Full color 3D scanning and surface convolution over textured 3D models for transradial prosthetic socket design and manufacturing

Leaf Oliver Preston Thorseth

Thesis submitted for the degree of
Master in Informatics: Robotics and Intelligent Systems
60 credits

Institute for Informatics
Faculty of mathematics and natural sciences

UNIVERSITY OF OSLO

Spring 2022

Full color 3D scanning and surface convolution over textured 3D models for transradial prosthetic socket design and manufacturing

Leaf Oliver Preston Thorseth

Full color 3D scanning and surface convolution over textured 3D models for transradial prosthetic socket design and manufacturing

http://www.duo.uio.no/

Printed: Reprosentralen, University of Oslo

Preface

Since my early teens I have been fascinated by the concept of machines assisting humans with physical movement. Not only are robotic exosuits and advanced prosthetics staples of some of my favourite fictional universes, such as the Aliens franchise or the Original Cyberpunk RPG, I feel like I have been lucky enough to grow up seeing many of the visions of science fiction writers and futurists such as Michio Kaku become reality. Whereas the robotic exoskeletons and advanced prosthetic implants of science fiction are often used for war, or portrayed as a corrupting influence on humanity, I find that their real life counterparts are almost exclusively humanitarian, and contribute to giving those they assist a more worthy and fulfilling life. Whether it's exosuits being used in elderly care or rehabilitation, or prosthetic limbs giving people the ability to perform daily tasks, excel in sports or even simply boosting the users confidence and helping them deal with a traumatic experience. Part of the reason I applied to the ROBIN program at UiO in the first place, was the opportunity to work with devices like these. I got my first opportunity to do this in the first semester of my master's degree, where I was able to take part in a project with two other students and a postdoctoral researcher looking at recognising hand gestures, using high density electromyography. This project was eye opening and inspired me to propose a master's project involving high density electromyography and motion capture of the hand. However, based on the faculty's wish that we choose a pre-defined project, and realising that a prosthetic hand company had proposed a project to the university on digital socket customization and design, I jumped on the opportunity to take on this project.

As we were two students who had shown interest in the project proposal, we consulted with each other and our supervisor to each find our own unique approach and area of focus. My proposal was rather straightforward, I would familiarise myself with self-suspended transradial socket production and build on previous attempts at a digital design using improved scanning technology, and see if I was able to improve on the results of previous studies. The first shake up in the project came after the summer, when we realised that the company that had proposed the project, where I also had an external supervisor, had been working over the summer on many of the same things I had planned on doing. They were also quite reluctant to share their methods and results due to patent issues. This spurred me to change the focus of my project somewhat. My external supervisor expressed an interest in automation of the socket design process, so I proposed looking at image analysis of 3D scanned limbs, as this could potentially be applied to automation. Thinking about potential approaches to this I ended up proposing a novel method of applying 2D convolution to the surface of 3D objects. Implementing this method became a large focus of this project, experimenting and discussing with my supervisor when I was stuck and eventually coming up with a working implementation.

I was hoping to continue working with the prosthetics company mentioned earlier, and exchange ideas for the more directly prosthetic related aspects of the project. Unfortunately, they did not take any initiative to continue working

together on the project and we later found out that they had gone bankrupt.

Due to the double focus of the practical part of this project (working with a digital socket production workflow and developing a novel way of applying convolution), both of which I was highly motivated for and wanted to push as far as possible, I was forced to apply for an extension in order to carry out the necessary comparisons and evaluations, and finish documenting my work and reflections.

Though the bones of this thesis were laid out about 18 months ago, a great deal of it has been written in the last few weeks, and thanks to a trip planned the year before I even started my master's degree (and delayed until now by the COVID 19 pandemic) a great deal of it has been written on trains, busses, in the guest rooms of family and cheap hotels. But as I sit now typing up the finishing touches on a ferry crossing the North Sea, I am at least happy with the work I have been able to do.

Acknowledgements

I would like to thank my supervisor Mats Høvin for excellent input and feedback, for his honest outlook and motivating enthusiasm (as well as occasionally letting me use him as a very overqualified rubber duck). I would like to thank Nikolai for collaborating with me in the scanning, and volunteering as a test subject. Vilde also has my thanks for acting as a test subject and being incredibly supportive despite having to sleep many nights close to my bright screen and loud keyboard. I am grateful to the ROBIN engineer Vegard for ensuring I had access to the technology that made this project possible (and for properly instructing me so I didn't break any of it). Finally, I would like to thank the generous people at the upper-limb department of Sophies Minde Ortopedi for lending their invaluable expertise and insight to this project.

Contents

List of terms

Prosthetic/anatomical terms

- (Prosthetic) socket: The interface between the amputee's residual limb and the prosthetic attachment (such as a hand or foot).
- Transradial: Between the elbow and the wrist.
- Self suspending socket: A socket that is suspended without the use of straps, vacuum, a liner or other suspension aids.
- Medial: towards the middle of the body.
- Lateral: Away from the body.
- Anterior: Facing forward.
- Posterior: Facing Backward.
- Epicondyles: The medial and lateral protrusions of bone making up the upper part of the elbow joint.
- Olecranon: The sharp bony protrusion that juts out below the elbow epicondyles.
- Cubital region: The anterior side of the elbow joint, between the bicep and the lower arm.
- Anatomically neutral position: The position in which the human body is usually presented in a medical context. Similar to the position of a cadaver on an autopsy table, except standing upright.

Genlib variable names/terms used when explaining code

- vG: voxel grid - 3D array representation of a voxel object including empty voxels.
- vS: voxel sparse - sparse representation of a voxel object, holding the coordinates of non-zero voxels in a n*3 array.

- vGlocal: a local subsection of vG.
- vGsolid: vG but we ensure that the object is solid.
- Surface voxel: a voxel that is not fully encapsulated by other voxels.
- vGshell: vG but we ensure that the object only contains surface voxels.
- Global unit vectors: the three unit vectors of the axes of the coordinate system defined by vG.
- Kernel/kernel grid: an n*n grid representing a 2D convolution kernel.
- Kernel unit vectors: the two unit vectors of the coordinate system defined by the kernel grid.

1 Introduction

As a non-prosthetics user (particularly one involved in STEM), it is easy to imagine that we could be living in a cyberpunk-esque future where prosthetics perform just as well, or perhaps better than organic limbs. The technology seems to be there; high density EMG combined with deep neural network-based pattern recognition can accurately recognise dozens of distinct gestures[1], 3D scanners capable of accurately capturing subtle geometry are now in the pockets of teenagers across the world[2], CAD- and 3D-sculpting-software is easier to use than ever (and often free)[3, 4], and 3D printers can create everything from housing[5], to guns[6], to organic tissue[7]. However, in reality many people with limb differences choose not to use prosthetics at all[8], and amongst those who do, most use simple devices that often rely on decades old production methods, and working with several different companies over a long period of time, just to end up with an expensive device that often only provides basic functionality[8, 9, 10]. Though in some circumstances prosthetics can in fact outperform organic limbs, this is not the case for upper limb prosthetics due to the complexity of the human hand and the tasks that the hand is regularly used for[11]. We must ask ourselves then: Why are prosthetic limbs, especially arm prosthetics, seemingly so far behind the technology curve, and (how) can we use these new technologies to benefit prosthetics users.

Obviously this is a complicated question with many different answers, but fundamentally it may come down to this: obtaining and using prosthetics should not interfere with the user's life more than absolutely necessary. A prohibitively expensive prosthetic, a prosthetic that requires frequent visits to a clinic, or frequent adjustment or calibration by the user is not desirable. By its very nature cutting edge technology is more expensive and less reliable than well established methods. However, it is not as if it's purely consumer choice that determines what prosthetics are available to the public. Any industry is resistant to major changes in production methods, especially those that might make certain skill sets, methods, or even entire companies obsolete, and the prosthetics

industry is no different. There are several smaller companies and individuals that have embraced direct-to-consumer prosthetics production, using a fully digital workflow[9], and many companies are combining digital techniques with traditional methods[12].

Circling back to the point about minimising the disruption of the prosthetic on the users life, It is likely that for digital production methods to fully enter the mainstream, the prosthetics produced must be as easy, comfortable and reliable in use and to obtain as those made with traditional methods, as well as having in-built advantages.

One might ask, if the traditional methods are so reliable, then why should we seek to replace them? The answer to this is quite simple; there are several potential advantages inherent to digital production that could make the process of obtaining a socket easier and cheaper for the user, simply by virtue of using a digital workflow.

- In a traditional prosthetic socket manufacturing process, if there is a desire to preserve a record of the limb or the socket for use in later prosthetics then it is necessary to keep a bulky plaster cast or physical socket at the clinic. In a digital workflow, preserving the model for later reference or alteration is a natural part of the process.

- Acquiring a traditionally produced socket typically requires 3 separate visits to a clinic[8]. With a digital workflow it is possible to obtain a model of the limb without physical contact, and therefore parts of, or even the entire process, can potentially be done remotely.

- Traditional methods rely on highly skilled prosthetists to construct and fit the prosthetic, and because of the individual nature of residual limbs and the different needs of consumers, it is hard to bring in much automation. With a fully digital workflow it opens up for the use of automated software to aid in the design and fitting. This could allow the prosthetist to use their time more effectively and produce better end products. Assuming such software is kept open-source, it can be shared worldwide and lead to even greater knowledge sharing.

- For most prosthetic users obtaining a complete device involves at least 3 separate companies[8, 10], each with different facilities and expertise, but also with different interests and a need to make profits. With a fully, or even partially digital workflow it is easier for one company to provide a complete device, as many of the techniques and facilities will overlap, for example if both the socket and parts of the hand are 3D printed.

In this project we examine the challenges and pitfalls in using digital methods for transradial prosthetic socket design, by reviewing the research that has already been done on the subject, but also through our own experimentation. In the interest of creating more tools that make the use of digital methods for prosthetics easier and more accessible, we also present a novel method of applying

2D convolution operators to the surface colour of textured 3D models. Through all of this we hope to contribute to answering the following question: How can rapidly emerging digital technologies such as 3D scanning and 3D printing best be used to design and produce better trans radial prosthetic sockets?

2 Background

2.1 Transradial prosthetic sockets

In order to understand the role that digital technologies might play in transradial prosthetic sockets, we must first understand the general design considerations and demands for those sockets, as well as the traditional methods mainly used today. We have gathered information on this subject based on research and review papers, as well as speaking to orthotic/prosthetic engineers at the upper limb department of Norway's largest prosthetics clinic (Sophies Minde Ortopedi AS).

Transradial, meaning below the elbow, indicates that the socket is intended for use by a patient who has an intact upper arm and elbow joint, but that their lower arm ends at some point before the wrist. Such a condition is referred to as a transradial amputation and can be congenital (present at birth) or caused later in life. Though lower limb prosthetic users are more common than their upper limb counterparts, transradial prosthetics are the most common form of upper limb prosthetic.

For all prosthetics there are two main criteria that must be met for the prosthetic to be usable. It must a) be comfortable enough to be worn for an extended period of time and b) be secure enough that it remains reliably attached during the activities it was designed for. Both of these criteria are highly individual. What suspension methods that can be considered comfortable varies greatly from patient to patient, not only depending on factors such as sensitive areas, scars, exposed nerves, perspiration, and the amount of soft tissue in the areas of the limb that interface with the socket, but also non-physiological factors such as pain tolerance, general sensitivity to tactile stimuli, simply what suspension methods the patient is used to, and practical factors such as the amount of range of motion (ROM) the patient expects, and for how long periods of time the patient expects to wear the socket. The demands for secure suspension also change based on the weight of the socket itself, and the weight of the prosthetic hand it will be attached to, as well as the activities that the patient wants to perform.

In addition to comfort and security there are a number of other factors that must be considered when designing and manufacturing a socket, including factors such as aesthetics, durability (both short term in terms of withstanding force, but also long term with regards to wear and tear), weight, bulk, ability to fit electrical components for a myoelectric hand prosthetic, and of course cost. Due to all these different considerations there is no gold standard optimal solution for transradial socket designs, but rather a number of different designs

and suspension methods that are used at any given time. Sang et al. have published an excellent review of various transradial prosthetic designs and suspension methods, with a focus on biomechanics. They cover not only multiple self suspending designs, but also suspension by vacuum and friction using a silicone liner or fully silicone socket. Regardless of the exact design of the socket, the general principles used to achieve suspension remain similar, relying on the application of pressure and friction to specific parts of the patient's anatomy in order to secure the socket.

Generally speaking, common self-suspending socket designs similar to the Muenster and the Northwestern-style sockets rely on placing pressure to areas of the arm around the Elbow joint in order to achieve suspension[13]. It is generally accepted that the socket should act as an interface to the wearers skeleton, rather than relying on soft-tissue compression for suspension[13]. With regards to transradial sockets the most relevant skeletal features are the epicondyles (the protrusions of bone either side of the elbow joint, they are part of the humerus) and the olecranon (often referred to colloquially as "the elbow", this is the bony protrusion at the back of the elbow joint and is part of the ulna). Generally speaking two types of pressure are applied in order to anchor the socket, medial-lateral (meaning inside-outside in this context) pressure is typically applied above the epicondyles, and posterior-anterior pressure (meaning back-front) pressure is applied over the olecranon, with the anterior pressure ideally being distributed over a wide region of the socket.

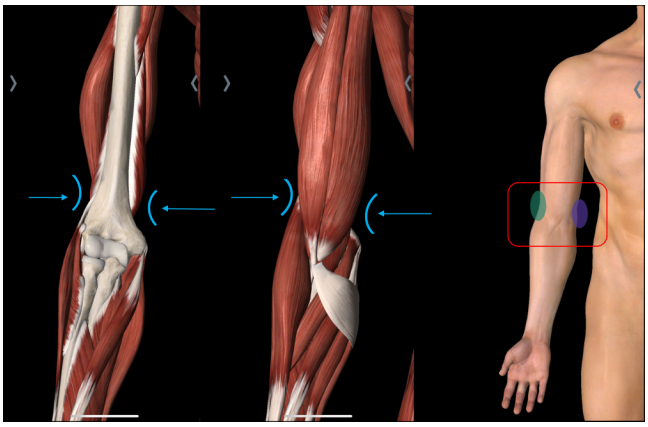

Figure 1: Illustration of where a socket would apply medial-lateral pressure. Image credit: Hy5

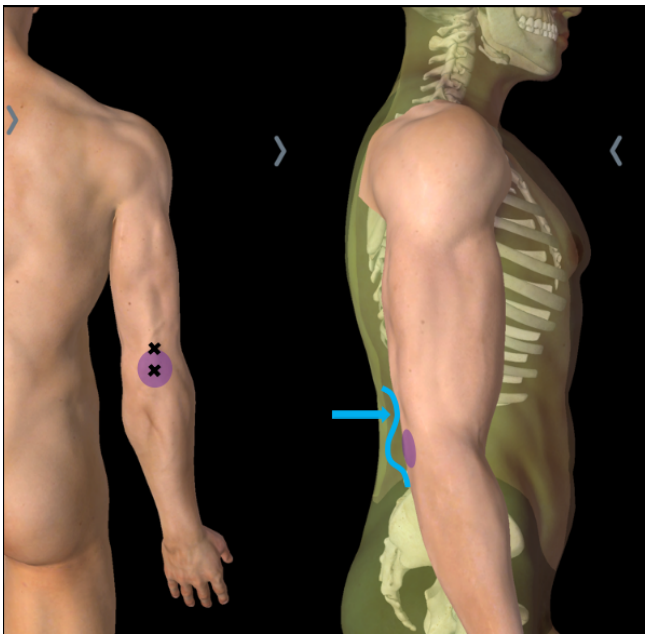

Figure 2: illustration of where a socket would apply posterior pressure, showing the olecranon region in purple, with the upper cross illustrating the correct position to apply pressure. Image credit: Hy5

If we imagine an extremely basic socket as simply a shell of uniform thickness that has an inner shape matching that of the limb. Such a shell could for instance be created by vacumforming a sheet of plastic around a positive cast of the limb, a process often used for test-sockets in modern prosthetic clinics[8]. This shell would then undergo a number of modifications in order to become a proper socket. The first type of modifications are what we have outlined above, bringing in certain parts of the socket in order to anchor it to the bones of the elbow, whereas the second type of modification involves cutting away parts of the shell in order to give the wearer more ROM when wearing the socket. The socket is trimmed above the epicondyles and olecranon on the back and sides of the socket, and this trimline often curves down below the cubital region on the front of the elbow joint, creating a semi-circular shape referred to as a smile-cut. The smile cut is important to reduce impingement in the cubital region when flexing the elbow joint.

In general a longer residual limb ("stump"), will allow for a larger smile cut, as there is more area further down the limb to give the socket leverage. General pressure and friction between the socket and the residual limb is an important factor in suspension, regardless of other suspension methods. In addition to the upper trimline, it is necessary to perform a cut in the socket around the olecranon, so it can move more freely during elbow flexion and extension. In some socket designs, such as the $\frac{3}{4}$ transradial socket a large section of the

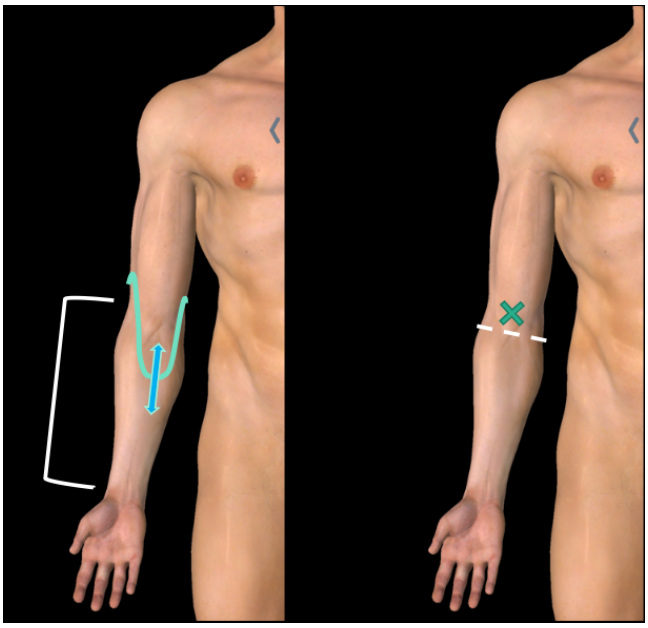

Figure 3: Illustration showing a smile-cut and the region that would be impinged without a smile cut. Image credit: Hy5

socket is omitted around and below the olecranon[13], but in other designs one simply makes a circular or oval hole of adequate size, so as not to further restrict ROM. A socket can also fully enclose the olecranon, but in this case the socket is brought out in said area instead, to relieve pressure and allow for freedom of movement. With regards to other suspension methods that do not rely on placing pressure around the bones of the elbow, some sockets are able to achieve suspension through vacuum suction. This can be done either via a silicone liner that connects to the socket body with a pin connector, or through direct vacuum, where the socket itself, typically a silicone socket, is able to hold a vacuum between itself and the residual limb. These kinds of sockets obviously place less pressure on the skeleton, and are generally less restrictive with regards to ROM, but are generally only secure for patients with longer residual limbs, do not allow for any breathability[13]; creating a hot and humid environment, and can be much more difficult for the wearer to don and doff without assistance. There are also countless strapped suspension methods designs, particularly popular in the DIY prosthetic scene[10]. These rely on velcro or other fasteners and are either less secure, relying entirely on compression of the residual limb for suspension, or they are strapped above the elbow, necessitating a hinge or flexible connector and potentially causing impingement in the lower bicep and cubital region.

2.1.1 Traditional Prosthetics workflow

While the socket production workflow varies somewhat from clinic to clinic, and even prosthetist to prosthetist and case by case within the same clinic, the same general steps are followed in most cases and have been largely unchanged since the 1980s. Olsen et al. describe a traditional socket production workflow that matches what we have learnt in discussion with prosthetic engineers in Norway[8, p. 2-3]. To quote: ”

1. Patient History: Initial consultation with the patient to gather background information and identify the sensitive and painful areas of the limb.

2. Limb preparation: Marking areas of interest on the limb using an indelible pencil, either directly to the skin or over a thin sock, e.g. the olecranon, epicondyles, and any sensitive areas.

3. Limb shape capture: Wrapping the limb in plaster-soaked bandages, whilst the prosthetist moulds the plaster and applies pressure over areas that will assist with suspension and stability. The cast is allowed to dry and then removed to be filled with plaster to create a positive model of the limb.

4. Initial modification: The positive model is adjusted and smoothed by the prosthetist to ensure a correct socket fit and pressure distribution.

5. Diagnostic socket manufacturing: A transparent plastic sheet is vacuum formed around the model to create a diagnostic socket, aka test or check socket.

6. Further modification: The diagnostic socket is fitted to the wearer and checked for fit, comfort and suspension. Adjustments may be made using a heat gun or notes taken for further positive modelmodification.

7. Modified positive model creation: The socket is filled with plaster to create a final positive model of the limb.

8. Socket fabrication: The final socket is made, generally using lamination, where several layers of cotton, nyglass and other soft textiles are set with resin around the cast, or vacuum forming using a thermoplastic.

9. Final additions: The final socket is fitted to the patient and checked. A second lamination is applied on top of the first, which forms the outer layer of the prosthesis and adds functional details e.g. the wrist.

”

Additional observations and caveats: Steps five, six and seven are optional, but can be necessary to achieve a viable socket, and several iterations of test-fitting and modification are used.

The second lamination applied on top of the first may be called an outer socket and does not play a role in suspension, but as Olsen et. al. state adds the other functionality to the socket. If intended for use with a motorised hand

the socket must contain the necessary components for the hand to function. These are typically housed within the outer socket, however, myoelectric sensors must obviously be held within the inner socket and are placed at precise locations based on the patient's ability to constrict the muscles in their lower limb (typically two sensors are used).

It may be necessary for the inner socket to have an adjustable flap that can be tightened after donning if the patient's particular anatomy makes it difficult to don otherwise, however such modifications are time-consuming to manufacture.

Several of the steps in this process require passive time waiting for plaster or resin to cure, etc. Therefore, socket production is typically carried out over several days and several visits to the clinic for the patient. The engineers we spoke to had previously carried out same-day manufacture of a socket. However, this was generally seen as a waste of human resources due to the waiting times where the prosthetist would usually see other patients.

2.1.2 Current research and innovation

Though the methods used in most prosthetic clinics today have been largely unchanged for decades, there are still many dedicated researchers worldwide driving innovation, not only in improving the existing designs and workflow, but also looking at novel ways of socket manufacture.

Because of the proportionately larger number of lower limb amputees in the world, a larger proportion of the research into sockets (and prosthetics in general) is aimed at lower limb prosthetics[10]. Fortunately, much of the research done on lower limb sockets is also applicable to upper limb. However, there are some important caveats to be aware of, particularly in regards to suspension, and the forces that upper and lower limb sockets are likely to be subjected to. Though the legs and arms in many ways have equivalent anatomy (one large upper bone connected via its epicondyles to two lower bones), modern transtibial sockets are generally not anchored to the knee joint in the same way that transradial sockets are to the elbow. Instead they are more commonly suspended with the use of liners or vacuum[14]. The legs being larger than the arms also provides a larger area for the socket to apply compression and friction to, for the purposes of suspension. There is also the obvious difference in the movements expected by the upper and lower limbs. The upper limbs are often used for fine motor functions, or to pick things up, subjecting the socket to a pulling force, whereas the lower limbs are typically used for more gross motor movements, but also support the entire body when upright, meaning that lower limb prosthetics are frequently subjected to powerful pushing forces down into the socket. One reason liners are used so often for lower limb prosthetics compared to upper limb is simply that rolling a tight silicone liner on is much easier with two hands than with one.

Digital methods used in combination with traditional methods As established, the prosthetics community has been slow to adopt new technology for a variety of reasons. However, circumstances such as patients living in remote

areas or being unable to conduct hands-on examinations can hasten the uptake of new technology. A web-survey conducted during the covid-19 pandemic sought to investigate how widely adopted digital technologies had become, and also how they were used[12]. Though most of the survey respondents were from southeast Asia, several western countries were also represented. Of the prosthetists/orthotists surveyed, 44% of them used some form of digital technology in their practice. Of those, 74% had CAD/CAM facilities where they worked. Digital photography was by far the most widely used digital technology, but hardly fits in the same category as the other technologies being discussed. Beyond that, 3D scanning was the most widely used technology. Approximately half of the survey respondents who utilised digital technology would scan for a lower limb socket. In the survey, 3D printing saw far less use than 3D scanning, with no use for final prosthetic sockets as far as we can tell from the paper. This indicates that the 3D scans were incorporated as a shape capture method within an otherwise traditional workflow. The prosthetists we spoke to in Norway had very limited experience using 3D scanning and related technologies for prosthetics, but one of them described having used a similar process, where 3D scanning and 3D printing was used instead of the initial plaster casting of the limb, in an otherwise largely traditional workflow.

Even amongst the survey respondents who used digital technologies in their practice, a majority did not believe that it led to better fitting prosthetics. Among the respondents who did not use digital technology in their practice, the reasons they gave for the lack of adoption included cost (64%), and lack of awareness and skills (51%).

Though this survey is only representative for Singapore, where a vast majority of respondents came from, its findings are still relevant from a global perspective, especially due to the lack of similar reviews from other countries. The general takeaway that 3D scanning, as part of an otherwise traditional workflow, is the most practical and widely adopted form of emerging digital technology within prosthetics, seems to hold true internationally. This is despite the clear limitations of scanning as opposed to direct contact with the patient, and information that can only be obtained through physical contact. However, these limitations are such that to our knowledge, a majority of prosthetists still exclusively use hands-on methods, even in times where physical contact between prosthetist and patient carry increased risks.

Fully digital methods Though the majority of prosthetists who utilise digital technologies only do so as part of a largely non-digital process, some researchers and companies are pushing towards a fully digital workflow. The company Prosthetic Design Incorporated[15], has developed a specially designed 3D printer for lower limb socket manufacture, and are currently offering lower limb sockets produced with a fully digital workflow, with the exception of manual post-processing of the socket. They claim that their printed sockets can be manually modified just as easily as traditional sockets, using techniques such as grinding and heating to deform the plastic. They produce check sockets and

final sockets from the same material, and claim to have successfully provided hundreds of patients with 3D printed sockets over more than 7 years without a single socket failure. The company also uses 3D scanning and printing technology to reproduce sockets created with traditional methods, blending traditional and digital methods. We are unaware of any equivalent successes when it comes to 3D printed upper limb sockets. Several studies in recent years have attempted to create transradial prosthetic sockets using a fully digital workflow. Ismail et al.[16] and Olsen et al.[8] had relatively similar approaches, in that both wished to essentially recreate traditional methods, simply using digital limb capture and 3D printing instead of plaster casting and lamination. The Ismail et al. study only had one participant, but they simultaneously developed two sockets for this subject, one using traditional methods and the other using photogrammetry with a digital camera, CAD software (Meshmixer), and FDM 3D printing. The main evaluation metric for this study was the deviation between the traditionally manufactured socket and the digitally manufactured socket, and it was indicated that the digitally manufactured socket was not as good as the traditionally manufactured one, but was considered tolerable by the subject. They concluded that photogrammetry and 3D printing can be a viable method of transradial socket production and more economical than traditional methods.

The Olsen et. al. study was more focused on the biomechanics of transradial socket suspension and the necessary modifications around the elbow joint. The study also had more subjects, all of whom were experienced prosthetic users. They used a structured light 3D scanner for limb capture and had experienced CAD users working with a prosthetist to design the sockets, which were printed using FDM. The sockets were evaluated both for comfort and security, and though they were able to produce comfortable sockets, none of the sockets were ultimately sufficiently secure. The authors cited the need for clinical expertise in socket design and modification, regardless of the tools used, as well as only having a geometric scan of the residual limb, and thus being unable to utilise markings to identify anatomical features, as reasons for their failure in creating secure sockets. The only project we are aware of where fully digitally produced transradial sockets have been put into actual use is the Sierra Leone Project[17].This project is focused on providing 3D printed prosthetics and orthosis for low-income communities in Sierra Leone, and with regards to upper limb prosthetics they have been focused on cosmetic prosthetics, where suspension is less of a factor. Nevertheless, their prosthetics have been used by people in their day-to-day lives, and have been shown to improve general quality of life in later follow ups[18]. The Sierra Leone Project also uses a structured light scanner and FDM 3D printing.

Moving away from trying to emulate the traditional workflow, one of the main appeals to us about digital socket production is the possibility for more easily producing adjustable prosthetics. In their Master Thesis, Abdelaziz developed and tested a transradial socket design that could be 3D printed and allowed for volume adjustment of the socket via a pair of adjustable wings[19], demonstrating that adjustable 3D printed sockets are viable. Despite the impressive engineering and design, the socket was not customised for the individ-

ual wearer, meaning that suspension had to be achieved through a simple strap around the upper arm. This lack of secure suspension was demonstrated in the human tests, where the socket would slip when subjected to a pulling force of 0.5kg.

With regards to automatically designed 3D printable upper limb sockets, Gorski et. al. have developed a system for automatically customising an upper limb socket based on a 3D scan[20]. The study compares two socket designs, a vacuum socket and an open socket, that are both automatically fitted for the subject based on a 3D scan with a structured light scanner, and 3D printed with FDM before being tested on the subject. The sockets are also put through a number of extensive durability tests in order to test the strength of the material itself. With regards to patient testing, the vacuum socket was found to be quite bulky and uncomfortable due to it being fully enclosing, whereas the open socket design failed to achieve secure suspension, though both socket designs seemed to fit well.

Overall we have been unable to find any literature describing fully successful self-suspending transradial prosthetic sockets produced with digital methods.

2.2 Digital limb capture

Shape capture of the limb seems to be the area where digital methods are closest to catching up with traditional methods. This is despite the fact that digital surface scanning will always have an inherent disadvantage over traditional hands-on casting in that the prosthetist is able to feel information that could not be gained from only a surface scan. Conversely, internal imaging techniques that provide a 3D image of not only the outside geometry of the limb but also the inside can potentially provide more information than a traditional cast, but appear to see little use in prosthetics.

2.2.1 Internal Imaging Techniques

By internal imaging techniques we mean any technology that lets us see beyond the surface of the limb, giving us a dense 3D image, as opposed to the sparse model created by a surface scan.

Ultrasound scanning uses the reflection of sound waves as they pass between materials of different density to generate sub-surface images. It is most famously used during pregnancy to observe the baby in-utero, and is uniquely suited to this application, as the amniotic fluid creates a neutral background for the imaging, unlike tissue which shows up on the scanner. Because tissue shows up on the scanner in a way not-dissimilar to bone, it is more difficult to separate the two when looking at an ultrasound image, compared to an x-ray. Nevertheless, ultrasound has been used successfully to diagnose fractures[21], particularly in patients where x-ray is considered high risk. Ultrasound is also a relatively affordable technology, and completely safe when used correctly, so investigating ultrasound for use in socket manufacture has some promise. If subsurface

scanning proves highly beneficial for a digital socket production workflow, then ultrasound could be a reasonable candidate, simply due to its safety and relatively low cost.

X-ray and CT Though x-ray is still state of the art for capturing the state and position of bones, it can be harmful to the patient[22]. Therefore, despite the fact that x-ray already has applications in some kinds of orthosis and prosthetics production[23], we see it as an unlikely candidate for mainstream use in upper limb socket production. Our goal is to make the process more comfortable for the patient, not potentially harmful.

MRI is similar to ultrasound, in so far as it is a largely risk-free subsurface scanning method that can be used to examine both tissue and bones. In MRI a large magnet is used to create a strong magnetic field in and around the patient, then radio-waves are used to excite the water molecules in the patient's body, releasing energy that can be used to create an image. In general, MRI is very accurate and can detect some things better than x-rays, but has the disadvantage of requiring very large and very expensive equipment. Though a full MRI of a residual limb would no doubt be an amazing starting point for digitally designing a socket, MRI as it exists today is not practical to use in this context. Our goal must be to minimise the difficulty and cost of getting a custom socket, whilst maintaining quality of course. Requiring expensive procedures like MRI scans is counterproductive to that. If handheld MRI-like devices were to become the norm, or if our goal was simply to create the optimal socket regardless of cost, then MRI could be extremely relevant.

2.2.2 Photogrammetry and structured light scanning

By far the most common digital limb capture techniques in the literature are photogrammetry and structured light scanning, with structured light scanning being the most popular of the two[8, 21, 16, 17, 24].

Photogrammetry is the technique of capturing a 3D object from a series of 2D images, such as ordinary photos. These images are searched for identifiable features, the features are used to triangulate camera positions and create a point cloud, and the pointcloud can then be used to create a mesh. One interesting feature of photogrammetry is that it provides not only a 3D mesh, but it can always provide a texture for that mesh, using the original photos. Some of the disadvantages of photogrammetry is that it requires you to take a lot of photos from all different angles, and is relatively computationally intensive when running through said photos and reconstructing the 3D object. Generally, as long as one has an nvidia-enabled gpu one will be able to do it in a reasonable time with one's own computational power, but there are many different software solutions for carrying out the reconstruction, including some where you pay for cloud computation. Photogrammetry is also reliant on several factors to achieve a good result, such as flat lighting and an appropriate and consistent

focal length, not to mention the quality of the reconstruction software. However, photogrammetry is essentially free if you have a camera and a computer, and one can always texture (colourise) the model, as mentioned before. Photogrammetric models can be manually scaled using a reference that is present in the scan.

Structured light scanners are a type of active 3D scanner (others typically use lasers) that actively project a pattern of light in order to capture the geometry of an object. These come in many price categories, both handheld and stationary varieties, and even as iPad attachments[12]. These types of scanners perform very accurately on mat surfaces, with a lot of surface texture, and due to the active nature of their operation generally perform best in low lighting conditions. Under these ideal conditions they can accurately capture shapes and textures smaller than what the naked eye can see. Scaling of the scans can usually be done automatically.

Some 3D scanners can produce fully textured models, in addition to capturing geometry. In the case of the Einscan Pro 2X (Shining 3D) that we have at the ROBIN group, this requires an additional module that can be purchased for approximately 6000 NOK, a fraction of the cost of the scanner itself.

Both photogrammetry and structured light scanners have been shown to be able to accurately capture residual limbs, though the meshes produced by both methods usually require some manual cleanup when used for this application[8, 16, 17].

2.2.3 Image Processing for limb captures

One of our major areas of interest coming into this project is to explore image processing for textured 3D limb scans, in the hope that it might have applications in aiding in, or partially automating the socket design process.

To apply image processing to a limb capture, we must first understand the basic representations used for 3D objects. Both photogrammetry and structured light scanning first generate a point cloud, essentially raw 3D data consisting of observed points in space, these points are then meshed. Meshing refers to using the point cloud to create a new representation, this time not only with points in space (usually called vertices in this context), but with edges connecting them. The vertices interconnected by edges form polygonal faces which in turn form the surfaces of our 3D model. Common file formats for a mesh include STL which carries only geometric data, and OBJ (wavefront) which contains geometric information and can also include UV-indexes, used for applying texture to the model. To apply texture, or "colour", to a 3D mesh, the mesh must first be UV-unwrapped, meaning that the faces of the mesh are essentially folded out and placed onto a 2D image, so they can draw colour information from said image. Fortunately, our 3D scanning or photogrammetry software does this process for us, though the image textures and UV maps it generates are typically very complex, involving an image texture consisting of a vast amount of photographs of various sizes from various angles of the model, and a UV map where the faces are spread all over the various photos of the image texture.

In addition to point clouds and meshes, this project will be using voxel representations of 3D objects. Voxels are essentially 3D pixels, so when we convert a mesh to a voxel object we place our model in a discrete 3D grid, where every cube in the grid is either a zero voxel, meaning that it is empty, or a non-zero voxel meaning that it contains some part of our model. Though very intuitive, voxels have generally been quite unpopular as a 3D representation, because of the high memory cost of high definition voxel representations. 3D surface scans naturally produce sparse data, meaning that most of a voxel representation of a 3D scan will be empty. One advantage of voxels, is that many of the image processing techniques developed for 2D images translate directly over to voxels, simply with an added dimension, and the obvious difference that empty pixels are somewhat rare.

There are a host of different ways to interpret and process different 3D representations, and although we will be primarily concerned with voxel representations, we would like to bring up multi view fusion segmentation[25]. This can be used to classify various parts of a 3D object's surface, something we will be exploring in this project. Multi view simply refers to having a This can be used to classify various parts of a 3D object's surface, something we will be exploring in this project. Multi view simply refers to having a number of photographs or renders of the object from a variety of different known camera angles. Multi view segmentation takes these renders and uses them as inputs for a 2D segmentation algorithm which classifies different parts of the render. These 2D segmentations are then recombined and applied to the 3D model. There are also ways to do 3D segmentation while staying fully within the 3D realm, for instance with sparse 3D convolution[26]. Sparse simply refers to restricting the output to only include non-zero voxels where there are non-zero voxels in the input, thus retaining the shape of the object.

Both of these methods take into account the shape of the object when doing segmentation. However, when we attempt to do our own processing of the limb captures later in this project, we will be attempting to interpret pen-markings on the 3D surface, not the shape itself, thus what we will try to do falls somewhere between 3D and 2D image processing. During this project we use a slightly modified version of the Canny Edge detection algorithm. This algorithm was chosen because it is simple, robust, well understood and still in wide use today[27].

2.3 genLib

GenLib is a generative design library for Matlab developed by Mats Erling Høvin at the University of Oslo, Institute for Informatics. The focus of the library is to be a tool for rapid and dynamic research and experimentation, it is therefore implemented in a functional style without object orientation. This makes most genLib code easily readable and modifiable, especially when certain conventions, such as variable names, are upheld.

GenLib supports various representations of 3D objects, including voxel objects and various types of polygonal mesh. All of these representations are

essentially handled with arrays holding the positional and relational information of voxels, nodes , edges etc. For example, a voxel object is commonly represented by a 3-dimensional integer array of size x,y,z. Converting between representations is generally made simple by genLib functions. For this project an .obj file was imported into genlib with the importOBJtoPolyTriMeshFast function, the mesh from the .obj file was then transformed into an rgb voxel object using the polyTri2voxColor function, taking a texture image as an input, in addition to the outputs of the aforementioned import function. The polyTri2voxColor function was implemented during this project based on discussions we had about how .obj meshes are textured.

Because of our familiarity with genLib, Matlab was used almost exclusively for the programming needs of this project. Though it offers considerably less as a programming language compared to languages like Python or Java, the framework provided by genLib far outweighed those disadvantages. The methods discussed in this project do not rely on anything that is truly unique to Matlab or genlib, and are implemented in a transparent manner. They should therefore be straight-forward to implement in a different language. Some of the Matlab code written for this project will be available from a GitHub repository linked in the appendices. Because genlib is under development, a stable version of genLib was used throughout the majority of the project, this version will also be included in the repository.

3 3D scanning and 3D printing for upper limb prosthetics in practice

After the initial literature search we sought to familiarise ourselves with 3D scanning and printing specifically for socket production in a direct and practical manner. The goals for this part of the project can be broken into four main categories:

- Limb capture: familiarise ourselves with the various scanning options at our disposal, develop an effective procedure for acquiring limb-captures and acquire some high-quality limb captures for use in later parts of the project.
- Socket design: attempt to design some basic sockets from the limb captures, using 3D modelling/sculpting software and test them for comfort and security, in order to better understand the biomechanics and design-requirements for self suspending sockets, in a process similar to Olsen et al.'s[8] work.
- Printing: 3D print the sockets that were designed in the previous step and consider the difficulty of the print itself with regards to necaserry support and print time, as well as the strength and rigidity of the printed socket.

- Evaluate the comfort and suspension security of the printed sockets, hopefully providing further insight into the specific demands when digitally designing transradial sockets.

3.1 Limb capture

Early in the planning stage it was decided that we would work not only with a volumetric 3D scan but with surface texture (colour information) as well. Based on our early research into scanning methods we knew that this kind of fully textured model could be obtained both with photogrammetry, or with the active 3D scanner at my university department, an Einscan 3D Pro 2X (Shining 3D), if we acquired a separate colour-module for it. An Einscan 3D scanner was also used by Gorski et. al. to capture residual limbs in their automated upper limb socket design project. They did not report any major issues with the scanning process, indicating that the Einscan was suitable for this project also. We did some quick testing with photogrammetry, photographing an object of somewhat similar complexity and size to a residual limb, and processing the images in Meshroom to create a fully textured model.

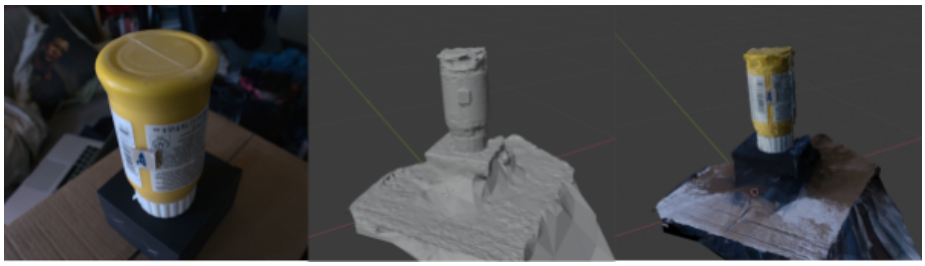

Figure 4: Our first attempt at photogrammetry. Left to right: one of the photographs used, untextured mesh, textured mesh.

As we can see the results are hardly perfect. We took in excess of 100 pictures, and only 43 of them were able to be used by meshroom to create the final model. Several studies have shown that photogrammetry can produce equally accurate, if not better results compared to structured light scanning[24], so we are certain that better results could have been achieved with more optimised camera settings, more experience and perhaps better software. Ismail et al. have shown that photogrammetry scans can be used to create sockets[16], so we could have gone down this route, however they also recommended an automated rig to help with the photography process (understandable considering the added difficulties of working with a patient instead of a static object). Most studies we examined agreed that the main selling point of structured light scanners was their reliability and ease of use[12, 24]. If we were to rely on photogrammetry for limb capture it would likely require a lot of additional labour and become a major focus/bottleneck in the process.

Fortunately, the department obtained a colour module for use with the Einscan, and we decided to abandon photogrammetry for the purposes of this project as it was not intended to be the focus of the project, and most, if not all, the other work we had planned would also transfer to working with photogrammetry-generated models.

We realised early in the project that human-trials, especially during covid and with a small pool of people to draw from such as transradial prosthetic users, were outside the scope of a masters project. As such, all of our work that required human subjects was done with a few students who volunteered on an un-official basis. In total 3 subjects were scanned, two male and one female, with BMI scores in the 18.7-27.5 range. As all subjects had fully developed and intact upper limbs certain workarounds were necessary in every stage of the project, but based on our literature review and conversations with biomechanical engineers and prosthetists we were confident that the foundational methods and principles would still transfer.

3.1.1 Scanning with the Einscan

When working with the Einscan we wanted to determine the optimal parameters for obtaining a suitable limb capture, as this was not something we were able to find extensive research on. Our parameters included the following:

- Tracking method: the Einscan software provides three options to track the object based on geometric features, based on texture, or based on a combination of geometric features and retroreflective markers placed on the subject.
- Lighting conditions: Our background research into 3D scanning indicated that an active 3D scanner would function best in dim-light, but we wished to see how much of a factor this was with the Einscan and also how this would affect the texture scan.
- Subject position: The subject must remain as still as possible during the scanning process, whilst also being as comfortable as possible. The relevant area of the limb must also be accessible to the scanner, and the limb must be held in a suitable position for socket manufacture.
- Operating technique: As we quickly realised, there is some skill required to obtain a high-quality limb capture, even with the other parameters being favourable. So developing a solid technique for physically operating the scanner in the specific case of limb capture was also necessary.

The main success criteria we considered when testing our various parameters were firstly to obtain a complete scan, in other words one that fully covered all the areas of interest (forearm, elbow joint and lower fourth of the upper arm) and secondly the loss (or rather the retainment) of tracking. Though the scanning software is usually capable of re-establishing tracking after a tracking loss, in all cases where this happened it resulted in a corrupted surface which would need to be repaired by hand in 3D modelling software.

Figure 5: Surface corruption in the interest area resulting from the loss and subsequent recovery of tracking.

Tracking and lighting In addition to our 3 tracking methods, we decided on 3 practical lighting conditions: Bright indoor lighting from fluorescent lights, somewhat dimmer indoor lighting, and near-total darkness. Though ideally we would have cross-tested every lighting condition with every tracking method, this would have required 9 scanning sessions for these tests alone (each taking up to an hour). Out of respect for the time of our volunteers we instead tested all 3 lighting conditions with feature based tracking and then tested the two other tracking methods with consistent lighting.

During these tests we would restart the scan if we had a serious loss of tracking and continue until we had a satisfactory result or could conclude that the method was not working adequately. In our lighting condition test, we achieved an acceptable result within 1-3 attempts for every lighting condition, leading us to conclude that this was not a critical factor. Further scans were therefore done in normal indoor lighting conditions as this was the most practical. Testing our tracking methods provided similar results: both feature based and texture based tracking provided good results within 1-3 attempts, in fact we experienced most tracking loss with the hybrid method using retroreflective markers, though this was almost certainly due to poor placement or inadequate density of the markers. Nevertheless, the hybrid method was deemed unnecessary and abandoned as it required additional setup time, and the tracking markers would need to

be removed in post-processing when using the scan to manufacture a socket. The feature based (geometric) tracking method was chosen for use in all further scanning.

Positioning of the subject and scanning technique As it would be necessary for the subject to remain as static as possible, all our limb captures were done with the subject seated and with their arm supported below the shoulder joint. Our initial scans were done with the subject holding their forearm at a 90 degree angle with regards to the elbow joint, as this was the position mentioned by Olsen et al.[8] and also the position where the bony features of the olecranon and epicondyles were most prominent on our volunteers. However upon further research we realised that we should perform the limb capture with the limb as close as possible to the standard anatomical position, as this is the position generally used by prosthetists, and would allow full access to all areas of interest. The scanner struggles to pick up enclosed areas such as between fingers, or the cubital region when the arm is held at a 90 degree angle. Obtaining a high-quality scan of a fully intact upper limb comes with additional challenges, not only because of the intricacy of the hand, but also because of the general length of the limb. As the focus of this project is on transradial prosthetics, the wrist joint and hand were not considered areas of interest when performing our limb captures. Because the scanner requires a certain distance from the subject to function, the limb has to be held out from the body during capture. This was done initially by the subject supporting their arm on the back of a chair with a pad for comfort, and later on a simple stand, 3D printed with a universal tripod attachment. Adequate support for the upper arm was necessary, as all subjects experienced some level of discomfort in that area during prolonged scanning.

The subject was instructed to externally rotate their arm, so the cubital region was facing upwards and to maintain a mostly straight, but not locked, elbow joint (175 degree angle between the upper and lower arm), and to be as relaxed as possible in the arm. All subjects chose to close their hand without clenching to help maintain a static position.

In order to avoid loss of tracking the two most significant factors were found to be large and constant overlap of the light pattern projected from the scanner and the surface of the limb, as well as maintaining correct distance between the scanner and the limb. Because it is essential for the subject to remain as static as possible, and for the scanner to maintain a right angle onto the surface it is scanning, the operator must essentially move the scanner in the perimeter of a cylinder with radius equal to 40cm, the correct operating distance of the scanner[28] around the subject's limb. This naturally presents some challenges to the movement of the operator performing the scan, especially in regards to scanning the underside of the limb, and in crossing over from one side of the limb to the other.

The projection from the scan is a rectangle, with a height equal to approximately 1.4 times the width. At the correct operating distance the projection from the scanner tends to be wider than the limb being scanned. At the start of

the project we were scanning with the manufacturer advised grip, holding the scanner in one hand by the handle, however we eventually realised that a more optimal grip for scanning a thin cylindrical object such as a limb was to hold the scanner in both hands by the sides, as one might hold a lunch tray.

When it comes to maintaining correct operating distance, the scanner software uses a visual cue to inform the operator whether the scanner is at a correct distance, too close, or too far. However, the operator is not realistically able to keep a monitor in view while manoeuvring around the subject and focusing on maintaining adequate overlap between the scanner projection and the limb. We found that the subject can easily monitor the visual indicator and call out changes in colour (green=good, yellow-red: too close, blue: too far), allowing the operator to adjust the scanner position. This may have the added benefit of engaging the subject and allowing them to focus on something other than the discomfort of maintaining a static position.

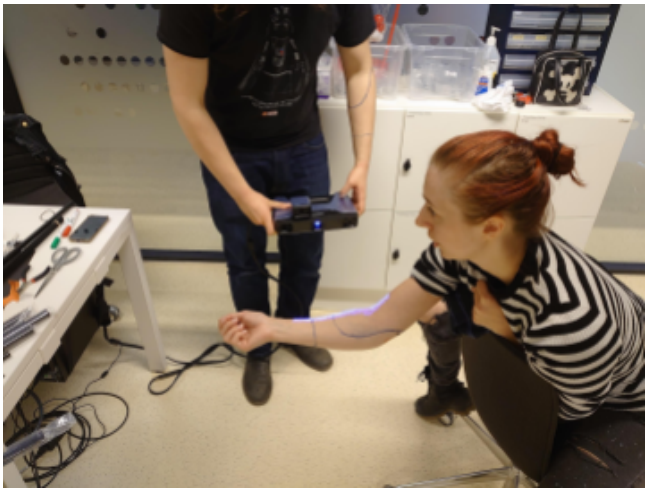

Figure 6: Operator scans side of limb while subject calls out change in distance.

Other considerations include making sure the power and signal cables are secured (we used a simple clamp), and ensuring that the operator has enough space and length of cable to safely manoeuvre around the subject. In addition the rapid flashing of the scanner projection caused discomfort in all three subjects and the operator, particularly when scanning the underside of the limb with the scanner facing upwards towards their faces. One potential volunteer subject was excluded due to photosensitive epilepsy.

3.2 CAD and 3D printing of rudimentary self suspended sockets

3.2.1 Designing a socket in Blender

Through the literature review and conversation with a biomechanical engineer working for a prosthetics company it was determined that traditional CAD software (Solidworks etc.) which is intended for precise mechanical engineering is generally not as suitable for design of prosthetic sockets[12], as this involves mostly organic shapes. Therefore Blender was used as the main cad program for this project. Blender is also the 3D sculpting program most familiar to us.

When designing sockets, the main goal was to become familiar with the process and challenges when creating self suspending prosthetic sockets with digital tools, not to create an optimal solution. An iterative approach was taken where several sockets were designed in succession for the same subject, with alterations done based on comfort, suspension and ROM.

The general workflow was as follows:

1. The fully textured model created by the scanner was imported into Blender.
2. A solidify modifier was applied to the model, in the positive direction, creating a shell around the limb.
3. The smile-line and supracondylar trim were made with a combination of boolean difference modifiers and separation of faces in the modelling environment.
4. A hole for the olecranon was created with a boolean difference modifier.
5. Sculpting was used to bring the surface in above the olecranon and epicondyles. The tools used were mainly the draw, inflate, scrape and flatten tools.
6. A boolean difference modifier was used to cut the shell approximately 6 cm above the wrist.
7. All modifiers were applied and the final model was exported as an stl for 3D printing.

The sixth step was only necessary for donning the socket as the subjects used had fully intact wrists and hands and would be omitted if this workflow was used to design a socket for an actual transradial amputee.

Early in the process we saw that identifying the olecranon and epicondyles from the scan alone was very challenging with regard to some subjects, depending on the anatomy of the joint as well as the amount of muscle, fat tissue and loose skin in the surrounding area. Therefore having scans with the anatomical features marked on the subject with a dry-erasable marker before scanning was extremely helpful.

The main factors for consideration were the placement and degree of indentation in the supracondylar regions and behind the olecranon, as well as the placement of trim lines.

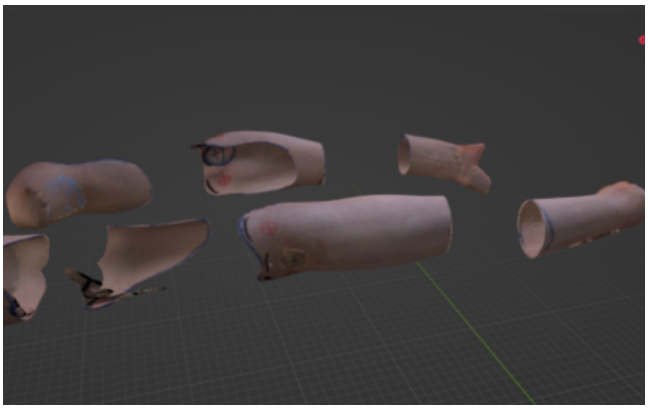

Figure 7: Two scans that have been solidified and trimmed. The central parts will become sockets.

3.2.2 Printing the sockets

An oft-cited motivation for the use of 3D scanning and 3D printing in socket production is the relatively low cost and greater availability of the tools and materials required to produce 3D printed sockets[8, 16, 12, 17]. Therefore it was desirable to work with a low-cost and widely available 3D printing technology. FDM was selected as it is by far the most widespread and economical technology, as well as being by far the most common in the studies we reviewed. The printing done for this project was largely done on a Fortus 250mc (Stratasys), which is an industrial-grade high-cost machine. However, the baseline technology and the properties of the end-products are comparable to those from much lower cost hobby level printers. The Fortus was largely chosen due to practical considerations to do with printers available at the university. However, it is also a highly reliable machine, and we would have been more likely to experience print failures due to nozzle clogging or poor bed-levelling if we were using a lower cost printer, which would only slow down the project, as the technicalities of 3D printing was not our focus.

The sockets were sliced in Stratasys Insight and printed with sparse infill and default support. The print times for a single socket varied between 12 and 18 hours, depending on the subject and the amount of support required for the design. In general most of our sockets required minimal support, only requiring some support in the areas where the socket had been brought in for suspension, creating overhangs. It is worth noting that an actual socket designed for an amputee would likely require more support, as our sockets could sit on the printer bed on the flat surface where they had been trimmed off to allow for donning by subjects with intact hands.

Once the sockets had been printed, the only post processing required was the removal of support material, which was made very easy thanks to the Fortus printer printing in a dedicated support material. This process would likely have

required more tools and some form of sanding if the socket was printed on a single extruder printer, however the end product would be very similar.

Figure 8: Different socket designs, furthest left: simple shell for testing the accuracy of the limb capture.

3.3 Results

Using the methods described in the section on scanning; holding the scanner by both sides and with the subject calling out distance warnings, high quality scans were obtained from all three subjects.

Having been properly instructed, one operator was able to scan without experiencing any loss of tracking (as defined by the Einscan software) on their second attempt using the scanner, while all 10 scans taken before adopting the two handed grip had some degree of tracking loss.

The final scans produced for each subject did not have any significant visible surface corruption that would require manual cleanup. However, it was later discovered when they were dissected in Blender that several scans had some amount of duplicate surface inside the model, which may have had a negative effect when using them in later phases of the project.

The sockets used in the iterative design process described above were not systematically tested but simply evaluated by the subject and designer for potential improvements before the next iteration.

At the end of this process two final sockets were created, one for the same subject that had been used during the whole iterative process, and one for a

new subject that had not had a socket designed for them before. These sockets were evaluated both for comfort and for security.

Both sockets had a similar issue where they could easily slip off during elbow flexion, and generally had poor contact with the elbow joint when the arm was held in a flexed position. We believe that this was due to the smile cut being too large, particularly in width. When the arm was held in a more neutral position, both sockets engaged properly with the epicondyles and remained secure when subjected to a pulling force of 2 kg. The sockets were not subjected to further testing of suspension as they were clearly not sufficiently secure.

Figure 9: Sockets being worn by the two test subjects. One socket had to be cut at the bottom to allow for donning and doffing comfortably.

Both sockets rated using the socket comfort scale (it is worth noting that none of the subjects had worn prosthetic sockets before and therefore had no grounds for comparison).

Figure 10: Socket comfort score

Socket/Subject	Comfort score (0-10)
Socket 1	3
Socket 2	7

Subject comments socket 1:

- Subject required help to don and doff the socket due to the relative size of their forearm and hand. This specific problem obviously wouldn't occur in an amputee. However, similar problems may occur depending on the shape of the residual limb, sometimes requiring a custom closing mechanism, according to one of the prosthetists we spoke to.
- Restricted ROM for elbow extension and forearm pronation. Subject has a hyper mobile elbow joint and is used to comfortably overextending their arm, which was not possible with the socket.
- Subject experienced discomfort when the socket was tested for suspension by pulling on it with a 2kg force, thus putting load on the epicondyles. Quote: "It felt like my elbow might sublocate if you pulled harder".

- The material felt hard and "scratchy", and the edges felt sharp.

Subject comments socket 2:

- Some chafing during movement, due to the rough material and lack of secure suspension.
- Restricted ROM for elbow flexion and forearm pronation.

4 Designing an Algorithm for 2D Convolution over a textured 3D surface

As established, prosthetists often use pen-markings to illustrate different areas or points of interest on a limb before creating a socket[8]. The general colouration of the skin can also give us useful information, such as areas of scarring. Thus, in working with digital scans for socket design, we wished to develop an algorithm that could allow a computer to process this surface-information, in order to assist in or partially automate the digital design process.

Pixel convolution is perhaps the most widespread and useful operation in both traditional image processing and in image analysis with machine learning. It allows us to do operations such as blurring or edge detection, and apply endless other filters that can be expressed with a weighted sum of pixel values. It also allows for feature extraction, and "condensing" of the image data, which are essential steps in modern computer vision[29]. Decades of research has been spent developing algorithms and methods for computers to understand pixel images using convolution. These algorithms often translate well to 3D[30], but understanding the textured surface of a 3D object is neither a fully 3D or 2D problem. This chapter describes how we developed a novel algorithm to apply 2D convolution kernels over the surface of a 3D object, and demonstrates its use by applying various classic convolution operations over said surface. The results are compared to methods of applying pure 3D and pure 2D convolution to the surface of a 3D object, namely sparse 3D convolution, and 2D convolution over the image texture.

4.1 Proposing a solution

When developing this algorithm there were two main directions we considered: working with the mesh and texture files directly, or voxelizing the object and working on the voxel representation. Through discussion we decided on the latter based on the tools we were familiar with, and general interest in voxel-based methods for CAD, as well as the fact that not all 3D point clouds can be easily meshed, and a method developed for voxels would likely transfer well to being used on point clouds. We will however also examine a mesh-based approach in the discussion section.

As voxels are analogous to 3D pixels, convolving through a voxel space with a 3D kernel is trivial, and applying a 2D kernel to the surface of a 3D object

is also somewhat intuitive: If the 3D object were a flat plane, 1 voxel thick, we could treat it exactly as a 2D image. However, as voxel surfaces become curved, irregular and differently oriented, this becomes a far more complex task.

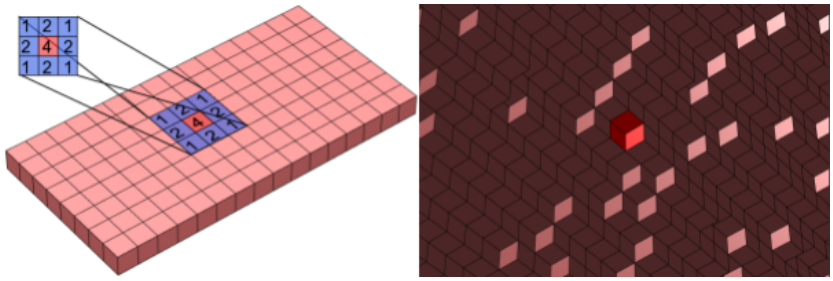

Figure 11: Left: A 3x3 2D kernel laid onto a simple voxel plane. Right: A complex voxel surface taken from a limb scan, oriented such that the highlighted voxel is close to the viewer.

The crux of this problem is placing a nxn convolution kernel over each surface voxel, in such a way that the centre value in the kernel overlaps the voxel in question, and each other value in the kernel overlaps a specific surface voxel that makes sense in relation to the centre voxel. To intuate how this kernel should be placed one could imagine taking a photograph of the 3D object, with the camera facing directly toward the target voxel. One could then draw a grid around said voxel to illustrate how each value should be placed, or in the case of a digital image, simply look at the pixel grid. Of course what we are doing when we take a photograph of a 3D object is essentially flattening it, so in a sense that is what we are also trying to do, but in such a way that we always choose the voxels that best represent how a human would view the surface, and do this for each surface voxel.

When trying to design an approach we decided to do initial development and testing with a 3x3 kernel, whilst keeping in mind that the method should be naturally expandable to larger kernel sizes.

4.1.1 Local neighbour search approach

When approaching the problem of placing the kernel, our first idea was a form of local search. The search would start in the target voxel, and search for neighbours in the surrounding 3D space using nested for-loops. Once a neighbour was found, we intended to add it to a sorted list of neighbours for the target voxel. If the list was full then it would be inserted only if it was determined to be a better fit than the least suitable neighbour already in the list. The reasoning for not simply adding every neighbour to the list is that unlike in a pixel image where each non-edge pixel has the same number of neighbours, we are essentially only dealing with edge voxels as we are convolving over the surface of an object. If we only consider first order neighbours (no more than 1 index away from the target voxel along any one axis), each surface voxel could

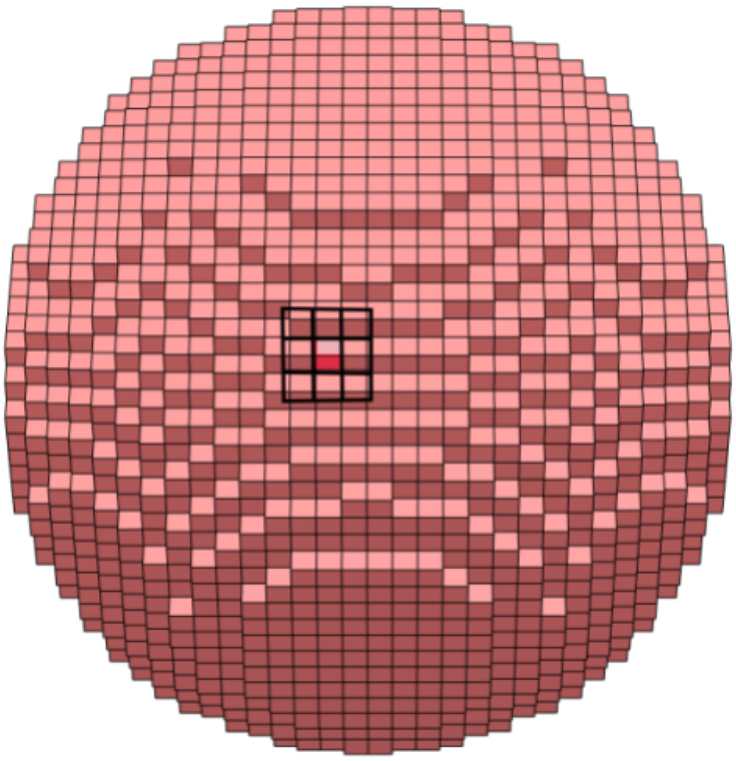

Figure 12: A low res. Voxel sphere with a grid drawn around the highlighted voxel. Because of perspective the surrounding voxels are not perfectly quadratic, but it is still easy to see which voxels occupy the majority of each square in the grid.

have anything between 1 and 25 neighbours (at 26 neighbours it would be fully encapsulated and could no longer be considered a surface voxel), as the surface of watertight voxel objects is often irregular and more than 1 voxel thick.

After implementing a voxel class with several methods to hold and sort neighbours, we ran some tests and realised several problems with this approach.

We were originally planning on using some form of padding when a voxel had less than 8 first order neighbours. In fact, we quickly realised that even with a 3x3 kernel, it would occasionally make sense to use non-first order neighbours for the convolution operation, because of the irregular nature of the surface.

Additionally, considering the neighbouring voxels one-by-one to decide whether they are appropriate does not make a lot of sense. The spatial information of the entire surface surrounding the centre voxel is required to place the kernel appropriately, since a neighbouring voxel's suitability relies not only on its proximity and angle to the target voxel, but also on the angles covered by the other chosen voxels. Our goal is to encircle the target voxel, so as to capture as much

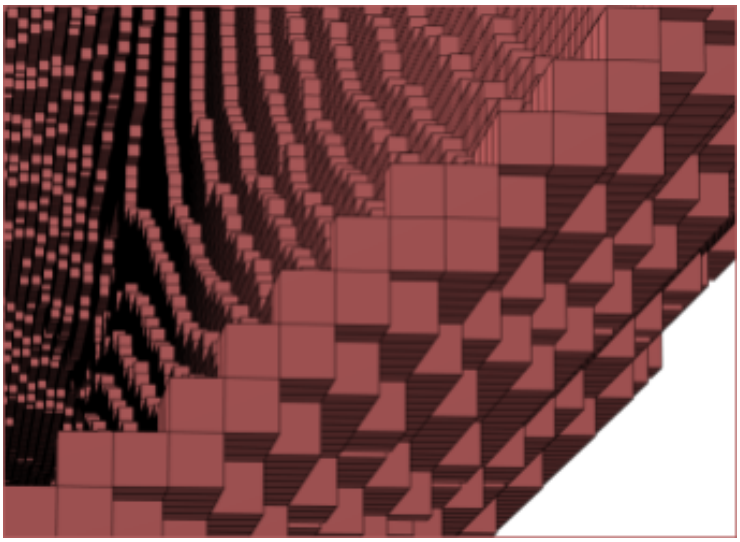

Figure 13: Cross-section of a voxel shell obtained from limb scan, showing the irregular nature of the surface.

of the variety in the surface around it as possible.

We theorise that if one was to follow this sort of approach, one would have to consider every voxel within kernel radius*steepness of the target voxel, where steepness is the expected maximal distance to the next voxel along any axis for a voxel that is one step away along a different axis. As an example, in the edge shown above the steepness would be two. When considering these voxels, one would have to optimise for maximum variation in angle (compared to the centre voxel), and minimum distance, whilst avoiding voxels directly "behind" our target voxel, in reference to the optimal viewing angle. Taking this into account we abandoned the idea of simply searching local voxels one by one, and decided to find a completely different approach.

4.1.2 Kernel projection

After considering the lacking aspects of our original solution, we went back to our intuition in order to find a new approach. Though our intuition is similar to a classical multi-view representation of 3D objects, rather than trying to flatten the 3D object, we are essentially trying to project a 2D kernel onto the surface so it becomes 3-dimensional. It occurred to us that we would need to find the orientation of the surface around the target voxel in order to properly orient the kernel being projected, and to determine the angle that it should be projected at. The orientation of the surface would essentially represent the natural viewing angle of a person wanting to examine that particular point on the object. Essentially, if we imagine a plane containing the centre of our target voxel, and oriented such that as much of the neighbouring surface as possible

is on one side of the plane, we would want our convolution kernel to have the same orientation as said plane.

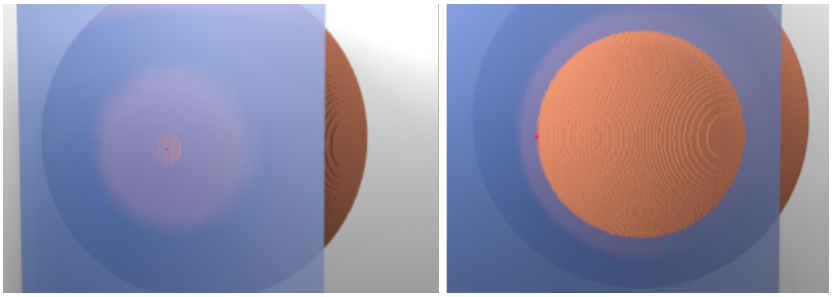

Figure 14: Left: Good placement of the kernel plane, right: Poor placement of the plane.

This seems like a good time to reiterate that as the purpose of this project is to develop methods that are applicable to prosthetic design, we will be assuming that the object we are working with is somewhat rounded overal (such as a residual limb). An object with very sharp edges or corners would obviously require different considerations and would be less intuitive to convolve over the surface of.

In order to find this plane that is aligned with the surface, we discussed the problem and realised that if we convert the voxel area that we are interested in into a sparse representation, where the voxels are represented by an array of 3D points, we could perform principal component analysis, and if the group of voxels formed a relatively flat surface, the two first component vectors should theoretically span our plane, with the 3rd component being perpendicular to said plane. We could then project the kernel matrix onto the surface, along the third principal component, and with the kernel oriented in the plane spanned by the first two principal components.

Figure 15: Based on these realisations the general approach can be laid out as pseudo code:

```
For each voxel V in the surface:
      vG_local = all voxels within radius R of V
      PrC = principal components of vG_local
      P = a point outside the surface, such that the vector from V to
P is equal to k*PrC[3]
      For x = -kernel_radius:kernel_radius
            For y = -kernel_radius:kernel_radius
                  Ignoring the case where x == y == 0
                  p_neighbour = P + x*PrC[1] + y*PrC[2]
                  while round(p_neighbour) is outside the surface
                        p_neighbour = p_neighbour + step_size*PrC[3]
                  kernel_placement[x,y] = round(p_neighbour)
```

We went on to implement this in practice. We did not go straight to working on a voxelized limb scan, but instead worked with a simpler shape: a high definition voxel sphere, when developing the basic algorithm. We will now detail several of the approaches we took.

4.1.3 Nested for-loop approach

The first approach we took was the one that seemed the most simple and intuitive. The target voxel was used as the centre point of a new voxel grid, vGlocal, with dimensions equal to twice the search radius variable, and we would iterate outwards along all three axes using 3 nested for-loops. In the innermost loop we would check if the point we were currently at satisfied the sphere equation given by the target voxel location and the search radius variable. $vGlocalSphere = (x - searchRadius)^2 + (y - searchRadius)^2 + (z - searchRadius)^2 =< searchRadius^2$

Having created vGlocal, we call the genLib function vG2vS which transforms the voxel grid into a sparse representation, an n*3 array of non-empty voxel indexes. With the voxel object in this form we can simply call the built-in matlab function for principal component analysis, in order to find the principal components of the surface surrounding our target voxel. We finished the rest of the program, iterating towards the surface and placing our kernel, square by square as the projection "hit" the surface. The program worked as expected when run for a single target voxel.

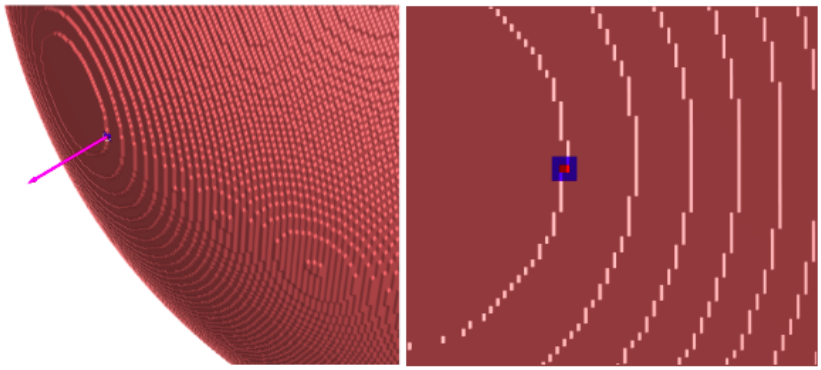

Figure 16: Image to the right shows the 3rd principal component based on the area surrounding our target voxel. Picture to the right shows the kernel projected onto the surface

We quickly expanded this solution to test on the entire sphere, containing over 2 million voxels, and discovered two things. Firstly, the algorithm was not always behaving as expected, in that sometimes it was failing to find a suitable neighbour for each square in the kernel grid, which should not be happening with a smooth sphere. Secondly, the program was extremely slow when run at this scale. We decided, perhaps foolishly, to put the first issue on hold, as it seemed

to be working as intended in most cases, and we were fairly confident that if we could first get the operation running fast enough to assure ourselves that we were working on something practical, we could later focus on generalising it completely and get it working as intended in all cases.

We set about timing various parts of the program and discovered that the crux of the computational cost per iteration was where we iterate along the 3 axes to create vGlocal, and where we convert vG local to a sparse form. Both operations are triple for-loops, so they both essentially take $O(searchRadius^3)$ time.

With this realisation we set about finding a faster method for creating a sparse representation of the local area surrounding the target voxel.

4.1.4 Fully vectorized approach

Based on the results of the previous method we discussed exactly what we were doing and we realised that the crux of the problem could be fully vectorised, removing the need for nested for-loops entirely, and hopefully making the program more efficient. Instead of isolating vGlocal in the grid domain, and then converting it into a sparse representation to find the principal components, we would simply operate in the sparse domain to begin with, using a series of vector operations on all of our non-zero voxels, in order to isolate the correct coordinates to be fed to the PCA. Essentially we converted the entire voxel grid to a sparse representation, before entering our per-voxel for-loop and the crux of the program, which had previously been two triple for-loops, was replaced with the following operations:

1. Find the distance between our target voxel and every other voxel in the space.
2. Create a logical array based on whether said distances are within the search radius.
3. Index vS using said logical array, leaving us with only the voxels that are within the search radius.

In matlab code these operations can be expressed as such:

$$circleVectors = vecnorm(vS - P, 2, 2);$$

$$within = circleVectors < searchRadius;$$

$$vSlocal = vS(within, :);$$

We first tested this new approach using a single target voxel, and after seeing it work as expected we moved to testing on the entire space. However, what we found was that this fully vectorised approach did not in fact save time with our current testing parameters. While the nested for-loop approach was running on approximately $O(2n)$ time, the fully vectorised approach is simply $O(kn)$, where k is a constant, in this case 3, the problem stems from n in the case of

our vectorised approach being the sum of all non-zero voxels, while in the case of our nested for-loop approach it is simply the chosen search radius. Since in these early tests the number of non-zero voxels was exponentially larger than the search radius, the vectorised approach performed worse. However, this approach and the others were reevaluated later in the process when the algorithm was tested on, and optimised for, limb captures.

4.1.5 Sphere filter approach

Based on the experience from the fully vectorised approach, we examined the idea of a hybrid solution. One idea was to create a template, a voxel sphere with radius equal to the search radius. This could be saved as a sparse representation, and then for each iteration moved, so it was centred on the target voxel using simple vector addition. I would then have to convert back to a grid representation in order to perform a logical AND operation to cut out only the voxels we are interested in. This method has several flaws, as it relies on multiple time-consuming operations, including the AND operation which was significantly time consuming when performed on something the size of vG, but we coded it up nonetheless, just to see how the idea behaved in practice. As expected it worked, but was the slowest solution so far.

Looking for faster solutions, we considered the possibility of simply indexing around our target voxel using the search radius like this: $vGlocal = vG[x + R : x - R, y + R : y - R, z + R : z - R]$ in order to create vG local. This is near instantaneous, but the drawback of this method is that it cuts out a cube that is not naturally aligned with the voxel surface, in other words it will behave differently depending on where it is placed along the object, and create cut-outs with angular corners. As expected this has a marked effect on the principal components of the out-cut area.

We discussed the various solutions we had worked on, and the problems with each, and we realised that the indexing method could be combined with a spherical template for little additional time cost. We would then create the template, consisting of a voxel sphere with radius R, before entering into the per voxel loop. We then index around our target voxel as above, but before converting to a sparse representation, we perform a logical AND between vGlocal and the sphere template.

We finished implementing this solution and saw that it worked as expected, and ran in about half the time of the nested for-loop method which had a similar methodology. Essentially it runs in $O(search_radius^3)$ time, as the vast majority of time consumption comes from converting vGlocal into a sparse representation. If we were to decrease the run time further we would have to examine the vG2vS function and see if it can be optimised somehow, or attempt to write an algorithm that performs PCA directly on the grid representation, without involving a vast time cost.

4.2 Ensuring expected behaviour

After being satisfied that the principles of what we are trying to do are sound, and having found a relatively time efficient method to extract the principal components of the surrounding surface for a target voxel, we proceeded to examine why the algorithm was occasionally "missing".

We found that which principal component projected out of the surface was not consistent. In the case of this high-resolution hollow sphere, both the 3rd and the 2nd principle component would occasionally be the "correct" component, with no clear visual pattern as to when one was selected over the other.

We also observed that changing the shell thickness of the voxel sphere had a dramatic impact on the number of misses, with a thicker shell leading to fewer misses. This indicated that part of the reason for the misses was that our projection was occasionally "passing through" the surface. We confirmed this by experimenting with decreasing the step size, and were surprised to find that this had a marked positive effect into very small step-sizes, so small that this iteration started having a noticeable effect on the runtime. After giving it some thought, this does make sense, since we might jump in between voxels if approaching close to the corner between them. We set the step size as small as was practical so it did not dominate the runtime and decided to examine the misaligned principal components first, as this seemed to be the more serious problem.

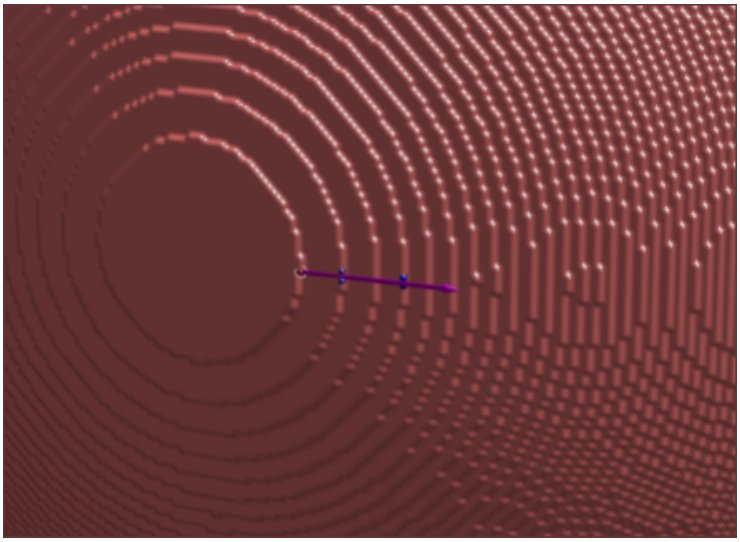

Figure 17: A case where the third principle component lies in the surface, resulting in most of the projected kernel missing the surface entirely, and the ones that didn't miss being placed being placed poorly. In this case the second principle component was normal to the surface and would have been preferable.

Since we are essentially looking for the component that is most directly

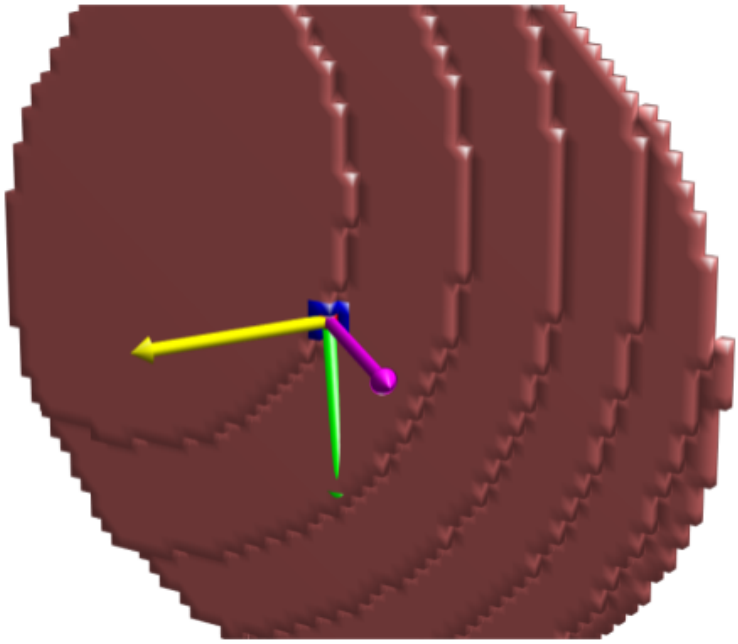

Figure 18: Example of the case where no principle component appears to be fully normal to the surface, only the part of the surface used to determine the principal components is shown. Principle components shown as arrows (green: first, yellow: second, purple: third). The blue voxels show the chosen placement of the kernel.

"out" from the surface, our first intuition was to examine how quickly we got out of the surface while going along each PC. Conversely we should move into the solid object when going backwards along a given component. In order to implement this kind of test we needed to ensure that we had a "solid" version of the object. To achieve this the solidify function from genLib was used, placing the startpoint at the centre of the voxel grid. We then defined a 5x5x2 grid, using the PC direction as the 3rd axis and the two other PCs as the first and second axis, placing said grid both in the negative and positive direction for each PC, starting from the target voxel. We then compared the number of non-zero voxels in both the positive direction grid, and the negative direction grid, taking the absolute difference. The principal component with the greatest difference was chosen. We could have simply gone 1 step in each direction, but because the PCs aren't necessarily perfectly aligned with the object this could easily lead to ties. Therefore we started with a 3x3x1 grid, but moved to a 5x5x2 grid and saw slightly increased performance. However, we still observed the occasional tie between PCs. We therefore decided to examine the possibility of using a combination of PCs as the projection vector. However, neither combining ties, or combining a scaled combination of two PCs based on their relative fitness

improved the performance.

Being satisfied with our method of PC selection, we ran a test of various search radius sizes, using an actual limb scan, tracking both the execution time, and the overall performance as measured by the number of total misses, and the number of "serious" misses, where we missed more than one neighbour during a single projection.

4.3 Using the mesh to determine the projection vector

In an attempt to evaluate the PCA based method for finding a projection vector, and to perhaps achieve better performance, we decided to implement a solution using the mesh representation of the object to determine the appropriate projection vector.

Since the 3D scanner produces triangular meshes which the voxels are generated from, in theory we would simply find a vector normal to the face that the voxel belongs to. In order to achieve this, we had to modify the genLib function polyTri2VoxColour which generates a 4 channel (volume,r,g,b) voxel representation from a triangular mesh. The function was modified so that each time it generates a voxel it also saves the index of the face the voxel was generated from, in a corresponding array. Because of the vast number of faces, and thus the vast range of possible indexes, it was necessary to use a large integer format to store these indexes. In this case uint32 was adequate. It was necessary to use a sparse representation, as the use of such a large integer type initially resulted in a crash due to excessive memory use. . The sparse method was based on the order created by the genLib function vG2vS.

A new version of the function for finding surface neighbours was then written. In this version the cross product of two of the vectors between face vertices was used to determine a vector normal to the face. In about 1% of cases the face vertices shared coordinates in such a way that the cross product was undefined, essentially also leaving the normal vector to the face undefined. This was circumvented by slightly shifting each face vertex a small amount along a different axis in those aforementioned cases.

Once a normal vector had been found for each voxel, it was sent to a function similar to the one written for PC selection. This function generated two more vectors to act as the unit vectors for our projected kernel. These vectors were generated by taking the cross product of the normal vector, and the unit vector of each of the two global 3D axis (xyz) it was least aligned with. As an example, if the normal vector was equal to [0.5,0.5,0.7] and thus mostly aligned with the z axis, we would take the cross products with the x and y axis to create unit vectors for the projected kernel grid. The theoretical reasoning for doing this was that firstly it might ensure fewer instances of missing a voxel due to coming in from an odd angle (say that the projection vector was almost perfectly aligned with the x unit vector, then the unit vectors for the kernel would be almost perfectly aligned with the global y and z vectors, creating a plane for the kernel that works well with the voxel space). Secondly, some amount of standardisation in the orientation of the kernel would be achieved with this method.

This method, using the mesh face orientations, was compared to the PCA based method. We also applied the logic of creating the unit vectors for the kernel via a cross product of the projection vector and the global unit vectors to the PCA based method. This way we could see if this performed better than using the remaining principal components as the kernel unit vectors, and thus also more fairly compare the PCA based and face-vector based approaches.

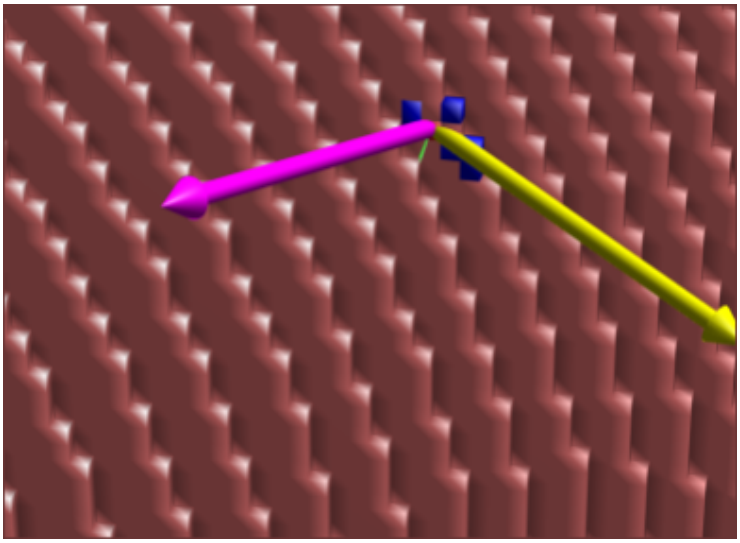

Figure 19: A serious miss. The kernel has been projected along the second principle component(yello) and the kernel voxels have not been chosen well, resulting in several of them being unassigned.

4.4 Image processing over the 3D surface

Now that the algorithm for placing the kernel was working to a satisfactory level, we had to design a representation from the results that could be used to perform convolution. Traditional ND-¿ND convolution is simple, fast, and uses no additional memory (simply the input array, kernel and output array). Whichever representation we choose should also be as memory efficient as possible, and should not involve any "look up" time. We had originally intended to implement a "voxel node" class, that would hold a 3D location and point to a number of neighbours, creating a directed graph. We also considered an array representation of neighbourship, using a non-zero-voxels by non-zero-voxels binary array. Both of these representations could help give an understanding of overall spatial relationships, as well as being helpful in evaluating, but the latter is both memory inefficient since we are dealing with a relatively sparse graph, and time inefficient since we would need to iterate through the columns in order to find the neighbours of a voxel. With regards to the former, we decided

that object orientation was ultimately unnecessary, as we could easily represent the necessary information as an array, and make use of the representations of the voxel space that we already had. Object orientation would also not be in keeping with the functional style we had used otherwise. This would of course be a sparse representation, so the length of the first dimension was equal to the number of non-zero voxels, with the second dimension being equal to the number of possible neighbours ($kernelDiameter^2 - 1$) and the third having a length of 3, in order to hold 3D coordinates of the neighbours. We could have had the 3rd dimension only hold a single index to the sparse voxel representation, however, this would have required looking them up during creation of the convolvable representation.

4.4.1 Implementing convolution

Implementing the convolution itself was very straightforward. We implemented a function that takes in an input and output array, as well as the convolvable representation and the sparse representation. It iterates through the convolvable representation and at each step it uses the convolvable representation to index into the sparse representation of vG, and uses those coordinates to index into the input array and create a kD by kD matrix of voxel intensity values.

If a coordinate value of 0,0,0 is found, the intensity value is left as undefined and a padding function is called to fill any empty values in the voxel value matrix. The current padding scheme is very simple, using the mean of first order neighbours to determine the missing value, and in the case that there are no first order neighbours, simply using the mean of all values in the matrix. Different padding schemes might be preferable depending on the convolution operator being used, but this simple padding was considered adequate for the purpose of the operations we tested in this project. Once any empty values have been filled, the constructed matrix is then multiplied with the convolution operator and the sum of the multiplied results becomes the value in the output array, exactly as with traditional 2D pixel convolution.

Before running a full edge detection algorithm the convolution process was verified using a simple gaussian blur and a sharpening operator.

$$(1/256) \times \begin{bmatrix} 1 & 4 & 6 & 4 & 1 \\ 4 & 16 & 24 & 16 & 4 \\ 6 & 24 & 36 & 24 & 6 \\ 4 & 16 & 24 & 16 & 4 \\ 1 & 4 & 6 & 4 & 1 \end{bmatrix}$$

(a) Approximate 5x5 gaussian blur operator

$$\begin{bmatrix} 0 & -1 & 0 \\ -1 & 5 & -1 \\ 0 & -1 & 0 \end{bmatrix}$$

(b) Simple sharpening operator

These operations were applied over each colour channel individually in order to produce the voxel objects pictured above.

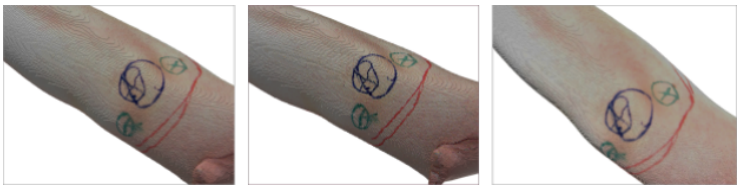

Figure 21: The results of these operations, applied to a 3D limb capture. Left to right: original model, sharpened, blurred.

4.4.2 Implementing the edge detector

As the convolution algorithm appeared to be working as expected, a full edge detection algorithm was implemented. The edge detection algorithm chosen was a version of the Canny edge detector, as it relies heavily on convolution operations, is simple, efficient and still in wide use today. The chosen implementation was as follows:

1. The rgb voxel grid was turned into a grayscale image by combining the values of all three colour channels. Both the NTSC formula and an equal ratio were tested, but they ultimately produced similar results.

2. A 5 by 5 gaussian blur was applied over the whole surface to remove noise which would produce small, unwanted edges.

3. A 5 by 5 sobel operator, consisting of two kernels, oriented in the kerlen x and kernel y directions was used to find gradients on the surface.

4. The gradient magnitude G was calculated as G = sqrt(Gx.^{2} + Gy.^{2}), where Gx and Gy were the outputs from the two kernels of the sobel operator.

5. The gradient direction was calculated as phi = arctan2(Gy,Gx).

6. The edge thinning step was then applied as per the canny algorithm, setting a given voxel's value to 3, if any of its first order neighbours along the gradient had a greater magnitude. This was implemented using the neighbours from the convolvable representation, as opposed to global neighbours.

7. Lastly hysteresis thresholding was applied, using an upper and lower threshold given as arguments to the canny edge detector function that was implemented.

Figure 22: Input model.

Figure 23: Greyscale model.

Figure 24: Gradient in "x" direction.

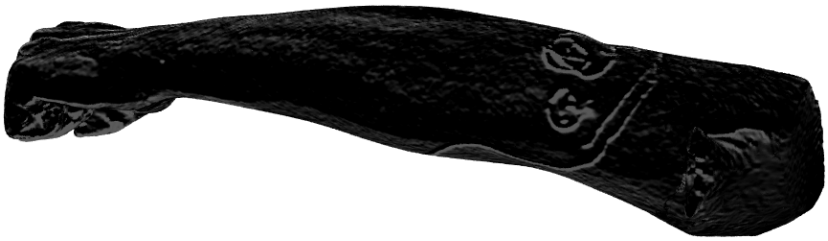

Figure 25: Gradient in "y" direction.

Figure 26: Gradient magnitude.

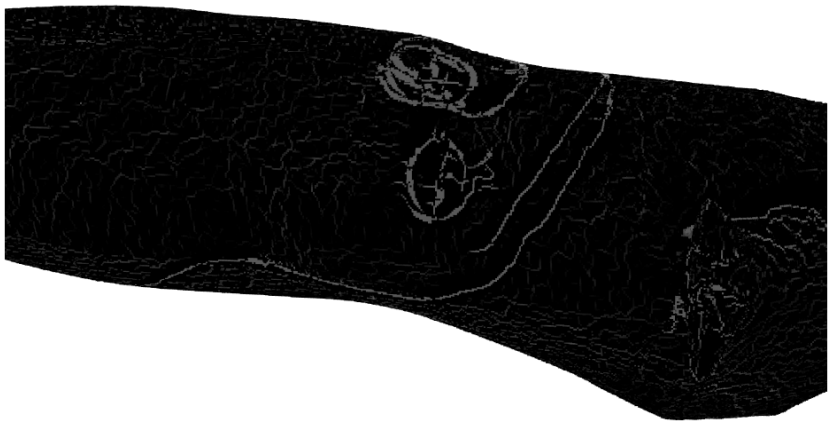

Figure 27: After edge thinning.

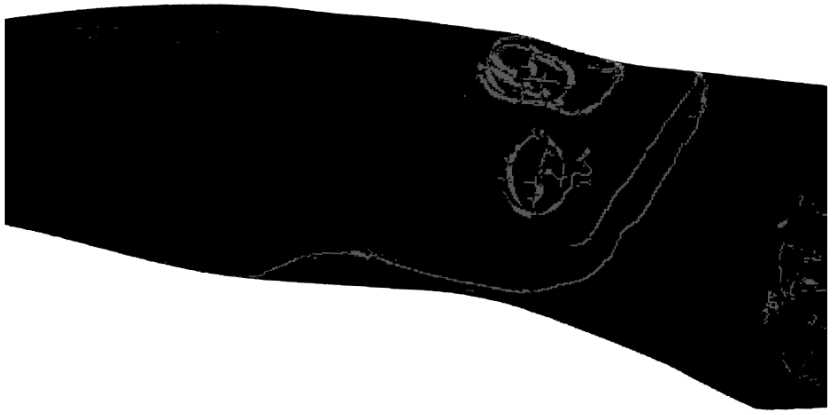

Figure 28: After first threshold.

Figure 29: Final output.

4.4.3 Colour thresholding and edge selection

As different coloured markers can be used to mark various features on the limb, one obvious desire is to separate voxels by colour thresholding. Another desire is to sort the edges from the edge detection algorithm by colour, and also to group the edge voxels into individual fully connected edges.

Colour thresholding The naive approach to colour thresholding is to simply set a single threshold for a given colour, say you wanted the blue parts of your voxel grid, you might say that you wanted all voxels where the blue value was over 150. In the case of our arm scans this would most likely leave you with some bright blue voxels, however, this method quickly breaks down. FIrstly, some colours might have a high blue rgb value, without appearing blue. Magenta for instance appears pink but has a maximal blue rgb value of 255. This issue was very apparent when we performed this kind of naive thresholding over the red channel, as the skin tone of our subject had a lot of red in it and was therefore retained alongside the red marker lines. Secondly, this simplistic threshold assumes that all your colours have a similar value, meaning that with this method any dark blue voxels would not be selected even if they were clearly blue to human eyes. Therefore, a comparative thresholding should be used. Instead of asking "how blue is this voxel?" we should be asking "how much more blue is it than it is green or red?". For the purposes of this project we simply classified a voxel as "blue" if the blue rgb value was greater than both the red and the green value, and vice versa for the two other colour channels. Thus voxels that had completely equal amounts of two or more of our main colours were rejected, but every other voxel could be classified as either red, green or blue. If we wished to threshold based on colours other than our primaries, the process would be more complicated, and we would either have a virtually infinite number of colours to categorise into, or we would threshold by how arbitrarily close a voxel is to a colour, probably using the ratios of the NTSC formula mentioned above.

These same takeaways can be applied for colour thresholding the edges that we get from our edge detection algorithm. Convolving over each colour channel individually is useful for simple convolution operations such as blurring, however, if we try to apply the canny edge detector to a single colour channel, the output will not be the edges that are our chosen colour. In fact, in our case, if we do this over the red channel we will not get our red lines at all, but we will get strong responses on the blue and green lines. This is because of the aforementioned fact that the skin colour has a lot of red in it, thus the transition between the skin colour and red line is very undramatic in the red channel. However the red line will show up if we run the canny algorithm over any of the other two channels, as here the transition between skin tone with a wide mix of colours and a pure red line is much more dramatic. In general, we need to take a similar approach when colour thresholding for edges as we did in the general case.

We can apply our colour thresholding to our model before we perform edge detection, effectively filling in voxels that do not meet our colour criteria with a local mean-colour value so that they don't contribute to creating edges. However, by doing this we are essentially artificially creating edges before we send our images to the edge detection algorithm, thus our edges will in reality be determined by colour thresholding, and not only by value changes in the image. Instead, we perform thresholding after running our edge detection on a grayscale version of the image.

If an edge voxel meets a colour criteria, or it has a first order neighbour that meets said criteria, then it is counted as being an edge of the corresponding colour. First order neighbours are included because an edge between two voxels, say a blue and a green, might fall on either side. This means that an edge will only be partially represented if we do not include neighbours. In this particular case, we could consider including second order neighbours, as we used a 5x5 sobel operator in our edge detection. However, including first order neighbours appeared to be sufficient.

However, one problem with this is that we will obviously have duplicate edges if we include first order neighbours for all 3 colour channels, especially for the background colour. To counteract this, thresholding using the criteria described above was done over the blue and the green channels. The rest of the surface was assumed to be majority red as this was the most prominent colour in the skin-tone of the subjects. Therefore the red edge voxels were defined as the edges that did not fit the blue or the green criteria.

Edge selection As the output from the canny edge detection algorithm is simply a grid of voxels showing which voxels are part of an edge, we must do some additional computation if we wish to group the edge voxels into individual edges. We define an individual edge as a group of edge voxels containing all edge voxels that can be reached by traversing first order neighbours, starting from any voxel in the group. Essentially we are looking for groups of edge voxels that are connected without gaps.

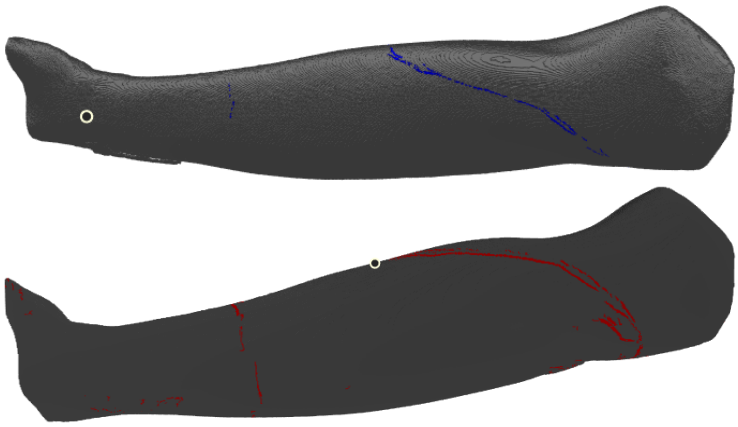

Figure 30: The problem with thresholding without including neighbours, showing blue and red thresholding. We can see that parts of the edge has been classified as blue, and parts of it as red.

To find these groups a simple recursive algorithm was implemented. A recursive function is called starting at an arbitrary edge voxel. The function marks the voxel it is called for as visited and then calls itself for all non-visited first order neighbours of the current voxel. It thus reccours until all voxels in the individual edge have been marked. We do this, starting at a new un-marked voxel until all voxels have been marked, and thus all edges are found. A sparse representation of the voxel grid was used for the recursion in order to minimise memory cost.

In this way the output of the edge detection algorithm was grouped into individual edges. This was also done for the specific groups created by thresholding as described above. The individual edges were sorted by length to help us choose specific edges.

Figure 31: Model with the top 20% of edges by length highlighted.

4.5 Results

Kernel Projection

One of the main challenges with this part of the project was finding a good angle to project our kernel from. For the PCA based method, a range of different search radii were compared. The upper boundary of 118 was selected for this comparison, as the smallest dimension of vG was 238 voxels long and a radius of 119 or more would give a diameter (including the centermost voxel) of over 238.

Figure 32: Accuracy for different PCA search radiuses, time use for sphere filter approach. For smile line scan, 682736 total surface voxels

Search Radius	Misses	Searious Misses	Execution time $*10^4$ seconds
3	846067	211494	0.0662
5	129430	32065	0.0979
10	3609	667	0.1617
20	2063	412	0.2819
30	1748	336	0.4169
40	1519	274	0.5649
50	1490	256	0.7312
60	1484	273	0.9493
80	1153	183	1.3927
100	983	147	2.0296
118	1045	172	2.8893

As we can see, the near-maximal search radius of 100 performed the best, with higher search radiuses dramatically out-performing small ones, however, having a larger time-cost when the sphere filter approach is taken.

4.5.1 Comparison of projection algorithms

Next the PCA based approach was compared to the mesh-face approach of finding a good projection vector. The best performing search radius from earlier of 100 voxels was used, and both methods were tested for 3x3 and 5x5 convolution kernels. A version of the PCA based algorithm using a cross product with the least parallel vG unit vectors as kernel unit vectors, rather than using the remaining principal components, was also tested in order to evaluate this separately.

Other parameters were adjusted to sensible values and then kept the same throughout testing. After completing the initial tests, we decided to vastly

increase the step size (the number of increments per voxel width unit when projecting toward the surface) for the best performing method, in order to see what effect this had on the number of misses.

Figure 33: For smile line scan, 682736 total surface voxels

Algorithm	Total misses	Searious misses
PCA-based 3x3, dist 20, step 100, sR 100	603	39
Face vector based 3x3, dist 20, step 100	6462	1473
PCA-based 5x5, dist 20, step 100, sR 100	6299	697
Face vector based 5x5, dist 20, step 100	39235	5575
PCA-based 5x5, dist 20, step 100, sR 100, xyz aligned	5478	450
PCA-based 5x5, dist 20, step 1000, sR 100, xyz aligned	1360	220

As the table shows, when all else was equal, the PCA based method with optimal search radius performed far better than the face vector based method.

Using the cross product with vG unit vectors to generate kernel unit vectors did show improved performance for this voxel object.

Increasing the step (number of steps per voxel size) ten fold had a large effect on the number of misses/serious misses, however it also had a dramatic effect on computation cost. Based on these results, an optimised approach for the kernel projection algorithm clearly differs somewhat from the original approach proposed in 3.1.2 and can be formulated in pseudo code as such (be aware that this pseudo code uses the matlab convention of indexes starting at 1, rather than at 0):

```
Function get_vectors(Voxel,PCs)
      for each pc in PCs:
            pos = 0
            for x = 1:2
                  for y = -2:2
                        for z = -2:2
                              xvec = x*pc
                              yvec = y*remainingPCs[1]
                              zvec = z*remainingPCs[2]
                              pos=pos+vG_solid[round(xvec,yvec,zvec)]
            neg = equivalent to pos but in the negative direction
            diff = abs(pos-neg)
      preffered_PC = the PC with the highest diff, multiplied by -1
if neg was higher than pos for this PC
      remaining_units = the two global unit vectors that are least
contributing to preferred_PC
      kernel_unit1 = norm(cross(preffered_PC,remaining_units[1]))
      Kernel_unit2 = norm(cross(preffered_PC,remaining_units[2]))
      return preffered_PC,kernel_unit1,kernel_unit2


For each voxel V in the surface:
      vG_local = all voxels within radius R of V
      PCs = principal components of vG_local
      prefPC, kernel_unit1, kernel_unit2 = get_vectors(PCs,V)
      P = a point outside the surface, such that the vector from P to
V is equal to k*prefPC
      For x = -kernel_radius:kernel_radius
            For y = -kernel_radius:kernel_radius
                  Ignoring the case where x == y == 0
                  p_neighbour = P + x*kernel_unit1 + y*kernel_unit2
                  while round(p_neighbour) is outside the surface
                        p_neighbour = p_neighbour + step_size*PrC[3]
                  kernel_placement[x,y] = round(p_neighbour)
```

4.5.2 Visual evaluation

As a comparison to the novel surface convolution method presented here, we tried a naive approach where the Canny edge detector was applied to the image texture jpg, before it was mapped to the scan. The open-cv implementation of canny was used and the threshold parameters were manually tuned until good results were achieved on the image. However, we see that the way the model is UV-wrapped means that many of the edges are only partially transferred when canny is applied to the texture image, and not the model itself. We also see a lot of double edges, though this is likely due to the relatively high resolution of the jpg compared to the voxel object. Another simple alternative would be to use sparse 3D convolution. By sparse convolution we mean that the output is restricted to only including non-zero voxels where there are non-zero voxels in the input. This method has shown great promise for working with naturally sparse 3D objects created by surface scanning[26]. It is generally applied to understanding the spatial features of such objects in true 3D, however, by performing a specific type of padding we were able to force it to only account for changes in surface voxel colour, and not for the shape of the voxel object itself. A 3D sobel operator was used, finding a gradient in all 3 unit directions (x,y,z), and then finding the approximate total gradient magnitude G = sqrt(x^2+y^2+z^2).

Figure 34: Alternatives to surface convolution. From the top: Canny applied to the texture image, 3D sobel operator with and without padding.

In order to not simply detect the changes in shape a padding scheme was used during convolution where 0-value voxels were given a value equal to the value of the voxel opposite them, mirrored around the centre voxel. The result of this operation can be seen above. This is the raw output from the sobel operator, so no edge responses have been thinned or filtered away. If the sobel operator is used with sparse 3D convolution without padding in this way then the output is completely dominated by the edges between 0 and non-zero voxels.

As we can see, the novel surface convolution operation performs the task of edge detection over the texture values of a 3D object very well compared to the other more simple methods. The entire image processing workflow from .obj file to grouped edges was evaluated using two scans from two different subjects. These scans were not used during development of any of the algorithms and the

Figure 35: Canny implemented with actual surface convolution.

Figure 36: Source model.

parameters used, and were taken during a separate capture session. The scans used were visually inspected and determined to be suitable, and no manual processing was done before testing.

When evaluating single convolutions the results appear very promising, both the gaussian blur operator and each individual sobel kernel produced outputs that correspond well to what we would expect. We do see some large disruption of the gradient response in specific areas, when dissecting the meshes in blender later. These areas were found to contain duplicate surfaces within the model which likely interfered with the kernel projection process.

When continuing the Canny edge detector process, we found that the gradient magnitude image is somewhat muddled in places, specifically some areas with many lines close together such as cross-markings.

The edge thinning step is perhaps the least successful, as it is destructive, breaking up previously continuous edges in some places, and in other places it fails to thin the edge at all. The other image processing steps were not reliant on the kernel projection and behaved as expected. Overall, a majority of edges were detected and categorised by colour with some success. However, the edges were prone to gaps and we were therefore unable to categorise continuous edges by size without implementing other techniques to fill in the edges. Because of the inconsistencies with the detected edges, we were unable to directly apply the kernel-projection based surface convolution to generating prosthetic sockets within the limitations of this project.

5 Discussion

The main goal of this project has been to develop a broad understanding of how emerging technologies, primarily 3D-scanning and -printing might best be

used in the prosthetic industry, with a focus on self-suspending trans-radial prosthetic sockets. During the discovery phase it was found that much of the previous research that had been done on the subject was focused on the potential cost- and labour-saving aspects, and not as much on improving patient outcomes by aiding in the creation of a higher quality prosthetics[8, 12, 16].

Other than quality, one of the primary reasons sighted for the slow adoption of these technologies is the expensive licensing deals and courses required to access digital tools for prosthetic design[12, 9]. Several other studies also struggled with the lack of intersection between the comprehensive skill set and knowledge-base required to design and manufacture high quality prosthetics, and the skills required to use 3D scanners and CAD[8, 16]. In general, it appears that most research projects using 3D scanning/printing for prosthetic sockets are initiated by researchers with a computer science/technology background who consult with prosthetists and not the other way around. This seems a little strange given the amount of specialist training required to design and fit optimal prosthetics for different patients with differing anatomy and practical needs, compared to the relatively simple skills required to do 3D scanning and basic CAD - skills that are frequently acquired by hobbyists without formal training. This speaks to the general trend that the profession of prosthetics is focused on practice, with prosthetists largely focused on their clinical practice, with little or no time left over for research.

In speaking with prosthetic engineers, their general attitude has been that they would like to do more research, but that they are generally busy with their practices and have had mixed experiences when collaborating with universities/researchers. They also expressed that many researchers have come to them quite late in the process, and that they would like to be involved from earlier on in the process, even from the conceptual stage.

Regardless of how one looks at it, prosthetic users have highly individual needs, that require equally individual solutions if those solutions are to be optimal. The goal of this project was therefore not to find ways to eliminate the human element from prosthetics design, but rather to explore how digital scanning and 3D printing can assist in the process and make it easier to design and produce high quality self-suspending prosthetic sockets, bringing in automation not to replace the human element, but to more efficiently transfer the knowledge and expertise of the prosthetist into the final product.

5.1 3D Scanning of upper limbs

In the first part of this project we familiarised ourselves with various scanning technologies, before moving forward with scanning upper limbs using an Einscan Pro 2X structured light 3D scanner. Early in the process we encountered difficulty in manoeuvring the scanner around the subject, while keeping it at a correct distance and alignment. This led to all of our early scans having tracking issues and several overlapping surfaces. However, after standardising our scanning technique and setup (as described in section 3.1.1) we saw far fewer tracking issues, even when the scanner was operated by someone without

practice.

Even without any loss of tracking (as defined by the scanner software giving a warning), all scans demonstrated minor issues such as slightly miss-aligned textures and/or surfaces, including in some cases additional unwanted surfaces hidden within the model itself. Because issues persist even without the scanning software giving any warnings, it is likely necessary for a trained person to carefully evaluate the scans before one is chosen, at least while using a similar setup and technology. Depending on how the scans are used it may also be necessary to do manual cleanup, particularly of the seams where surfaces are miss-aligned. We were able to reliably obtain scans where this clean up was unnecessary, however this did take multiple attempts even with proper technique, and different circumstances might make it necessary to work with an imperfect scan.

As these seams from miss-aligned surfaces are quite characteristic and clearly distinct from "correct" parts of the scan, there is certainly an opportunity to research methods for automatically removing them (either fully automatically or with a "point-and-click" type tool). Removing these seams using normal CAD software, or rather sculpting software, can be somewhat tricky as the mesh is often complicated and irregular in those areas. It is worth noting that traditional plaster casts can also require cleanup and correction, so this is not a problem unique to digital shape capture. However, prosthetic engineers and clinic technicians are already adept at working with plaster, so it is unrealistic to expect them to move to a different shape capture technique that would require them to learn new skills, unless it has clear advantages over traditional methods.

As observed by Yang et al.[**?**], there is no conclusive gold-standard for limb shape capture, and it is therefore hard to conclude exactly how accurate in shape and size the captures done for this project were. However, they were tested by producing shells and test sockets and we did not experience any problems with regards to the accuracy of the captures, either when it comes to shape accuracy or size accuracy.

With regards to the scanner itself it was chosen for this project primarily because it was already available to us at the university. This is fortunate as the Einscan is a precise, high cost industry grade scanner. Comparatively, photogrammetry is virtually free, requiring only a digital camera and a mid range desktop computer (or high performance laptop) to perform. This is obviously highly relevant if the main motivation for moving to digital methods is to reduce monetary costs. However, from our experience, photogrammetry is more labour intensive, and the resulting models require more extensive cleanup.

Overall, we consider ourselves successful in developing a scanning methodology such that high-quality limb capture can be achieved with a high degree of consistency, even when the scanning operator does not have extensive training with the scanner, but has simply been properly instructed in how to perform the scan. The methodology will of course differ somewhat depending on the model of scanner used, however the principals should transfer well to any real-time scanning method.

5.2 Designing and printing sockets

As detailed in section 3.2 the 3D scans were imported into Blender and used to create self-suspending test sockets. As mentioned, none of the volunteers involved in this project were actually transradial amputees, so the sockets were designed to accommodate a fully intact hand and wrist, by cutting the socket short some distance above the wrist. Whilst this had some effect on the scanning procedure (hands are challenging to capture and not necessary for the project so they were ignored, but longer limbs are also more challenging to capture well) and on the printing (having a flat surface on the bottom of the socket meant that less support was necessary for printing), this in general had an acceptably small effect on the main goal of this part of the project, which was to familiarise ourselves with socket suspension, and attempt to design self-suspending sockets in a fully digital manner, similar to the process used by Olsen et al.[8], except with fully textured scans.

When creating the first sockets it became clear why pen-markings are often used in traditional socket manufacturing, specifically to mark the anatomical features of the epicondyles and olecranon. Whilst these were easier to see on some of our subjects, on one of the subjects they were completely hidden when the arm was held in an anatomically neutral position. Having the texture module for the 3D scanner was critical here, as it allowed us to mark the limb and have those markings accurately transfer to the 3D model. Suggested trimlines were also marked up on the limb before scanning. The full process of modelling and sculpting the socket in Blender and printing it is described in 2.3 (though the methods could no doubt be improved by someone with a deeper knowledge of Blender). The results were ultimately that we were unable to successfully create fully secure sockets and only one of the two final sockets was considered sufficiently comfortable. It is worth mentioning that several test-sockets and adjustments are often necessary before a satisfactory socket is produced, even when an experienced prosthetist is working with traditional methods. Also when creating the final sockets, one of the subjects used had already had several practice sockets created for them, which somewhat reflects that scenario.

Our lack of success may be due to the limitations of the digital tools used, and our somewhat limited digital modelling and sculpting skills, but it is also likely that we would have had better end results if we had more general prosthetics knowledge and experience. We believe that simply having an experienced prosthetist mark features and sketch out trimlines on the limb before scanning would be likely to improve the results. A similar process could also help with the lack of skill overlap described earlier in relation to prosthetic skills and 3D sculpting/CAD. Many prosthetic clinics already employ technicians to relieve the prosthetic engineers workload in less technically critical parts of the job, such as the casting of plaster positives. It is possible that a CAD/digital sculpting technician could fill a similar role, creating digital test-sockets based on scans and the markings and instructions of an experienced prosthetist.

5.3 Surface convolution for image processing over a textured 3D limb scan

We have previously asserted that there is a lack of accessible CAD software that is suitable for digital socket design, as well as a lack of overlap between skill sets, and also that socket designs are highly individualised and based on both a theoretical foundation of biomechanics and also the practical experience of both the prosthetist and the patient, thus making fully automated design of an optimal socket extremely challenging. Based on these assertions we wished to explore how the human element of socket design could be transferred to a finished digital design with as much ease as possible. We decided to examine the pen-markings already commonly used by prosthetists and see how these could be interpreted by a computer, in other words how we can process and analyse images drawn over the surface of the 3D scan. As convolution is such a fundamental operation in image processing, it was a natural starting point. Based on our our literature review and knowledge of the field of image processing, we were aware of several excellent algorithms for analysing the surface of 3D objects, however we are unaware of any algorithms for processing only the textural/colour information of the surface, disregarding the 3D shape other than how the textures sit in relation to each other. For this reason a novel algorithm was developed, allowing us to convolve with a 2D kernel over the surface of 3D voxel objects, essentially using 2D image processing techniques over a 3D surface. The full process of developing the algorithm is outlined in chapter 3, but the basic principle used was to determine a "natural" viewing angle for a particular surface voxel, using the principal components of the volume of voxels in the surrounding surface. This viewing angle is then used to project a grid onto the voxel surface, centred on our target voxel. This grid is used to associate a number of neighbouring voxels with the target voxel in a particular order so that they can be used as 2D neighbours for the purposes of convolution. Ultimately this process proved quite computationally intensive, as many 3D operations are. Therefore a significant amount of this project was spent designing this algorithm to be as efficient as possible. One of the computational bottlenecks is the isolation of a certain area of the voxel surface which is transformed into a vector of 3D points so it can be run through PCA. Two algorithms were found to be computationally efficient at this, with one scaling linearly based on the size of the entire voxel grid (fully vectorised approach), and the other scaling cubically based on the diameter of the area used for generating the principal components (sphere filter approach). In our specific case we found a large search diameter to give the best results, and therefore the linearly scaling algorithm was most efficient for the optimal case. However, which algorithm is more efficient will vary from case to case with both being reasonable. It is also possible that a computationally efficient PCA algorithm could be written to work directly on a voxel grid, in which case the sphere filter approach would immediately become the most efficient algorithm in all cases, as the main computational cost lies in converting between voxel grid and sparse point-based representations. The second computational bottleneck was discovered later in the process where we found that increasing

the number of steps taken toward the surface, or rather decreasing the size of those steps, improved the accuracy beyond what we were expecting. This is due to the possibility of "missing" a voxel when we are moving towards the surface. The voxels are a discrete representation of a 3D object, but because we can be moving towards the surface at any 3D angle, the line that we move along becomes continuous, in that we could meet a new voxel at any point along the line. Since stepping along our line is a discrete operation, we can occasionally miss a voxel entirely if it only intersects with our line between two steps.

It is worth noting that though computationally intensive, the process of creating a convolvable representation of a 3D object only has to be done once and the representation can then be used for any number of convolution operators. Despite this, computation time is still a concern for clinical applications, especially if one considers same-day test sockets to be one of the potential advantages of digital design and additive manufacturing. Computation times are of course entirely dependent on the chosen parameters, the voxel resolution as well as the hardware available. The algorithm for creating a convolvable representation of the surface can be easily parallelized, allowing the operation to be run in under an hour with the parameters used in our final evaluation on a desktop computer with a 3.7 Ghz 6 core CPU (AMD Ryzen 5 5600X) and 32GB of RAM. Memory will always be a major concern when working with voxels, and we were operating close to our limits memory wise with the resolution used in the final tests. Memory had to be carefully considered when choosing data structures in all steps of the process. Even this resolution is somewhat rough, in that relatively extensive smoothing was required to remove rough voxel texture from the finished socket. Ideally the resolution should be such that the diameter of a voxel is equal to the chosen printing height, ensuring minimal loss of detail.

When visually evaluating the results of our surface convolution we found that simple operations such as blurring and sharpening worked well, performing exactly as expected. However, more relevant and complex operations such as the canny edge detector did not perform as well as desired. As far as we were able to see, the sobel filters used to find the gradient in the per kernel unit directions each found the gradients that they should. However, the combined gradient magnitude was somewhat noisy in areas with a high density of edges. Similarly the edge thinning step of the Canny edge detector, which does not rely on convolution per-se, but used the spatial relationships established by the convolvable representation, was highly destructive, thinning most of the edges but also removing essential edge voxels and thus splitting edges in an unwanted manner. Further image processing was carried out to demonstrate how the edge voxels could be classified by colour and grouped into connected edges, though none of these processes relied on surface convolution.

There are several challenges still to be solved with the surface convolution algorithm as it stands now.

- Firstly, the evaluation metrics used in this project - the number of "spaces" in our projected grid that fail to find a voxel, and visual inspection of convolution results are both somewhat flawed. Number of misses only

tells us when something has gone grievously wrong, it says nothing about how well chosen the voxels are for the spaces that actually do get filled. Visual inspection tells us about the quality of the chosen voxels, but is also largely subjective. Also, many convolution operators can appear to work well even with very poorly chosen neighbour voxels, especially operators that are rotationally symmetric.

- Secondly, as the projections are oriented based on the shape of the surface, there is little uniformity in orientation for these convolutions from voxel to voxel. Similarly, there is no 2D spatial relationship between the surface voxels, beyond the voxels that get included in the convolution for each specific voxel. This means that a principle such as convolutional stride cannot be used directly with this type of convolution and the convolutions are essentially performed in an arbitrary order depending on the target voxel's position in 3D space.

These challenges can certainly all be addressed, at least to some degree. It is theoretically possible to manually create a "gold standard" result for kernel projection, especially on a low-resolution scan, which could be used as a comparison. Other metrics could also be created, for instance one could track the number of cases where two voxels are included in each other's convolution grids in corresponding positions, as this would usually be a desirable outcome. Visual inspection can also be improved by standardising the operators and criteria used, for instance seeing how effectively certain noise is removed by blurring, or how effectively an edge is sharpened. With regards to orientation, the orientation was already somewhat standardised by using the cross product of the projection vector and global 3D unit vectors to determine the orientation of the projection. It is also possible that if some surface based spatial relationship, beyond which voxels belong to a given voxels convolution, could be established, then voxels close to each other could be forced to have similarly oriented kernel projections. Such a spatial relationship could be established by creating a directed graph describing the new neighbour relations established by the convolvable representation. This graph could also allow us to implement a form of stride in our surface convolutions. Overall there is a lot of potential for improvement with this novel method for convolving over the surface of a 3D object. Because of its limitations, we were unable to apply it to generating prosthetic sockets from markings on the surface, by automatically cutting trim lines, and placing indentations above the epicondyles etc. However, the method certainly showed promise overall, being able to perform traditionally 2D operations over a completely 3D surface, with far better results than performing 2D convolution on the texture image, or sparse 3D convolution with padding to remove geometric features.

This is not to say that there might not be better approaches to achieving the same result. It is certainly possible to work directly with the UV-map and convolve over the surface texture in that way. However, the UV maps for limb-scans are highly complex and themselves describe a 3D surface as opposed to a fully connected flat surface. The UV-map essentially consists of polygonal (in

this case triangular) faces of various sizes that may be connected to any number of other faces at the corners. Convolving within the borders of a single face in the mesh/UV-map would be trivial, but dealing with moving from one face to another and neighbouring pixels belonging to different faces is a whole new project in itself, though one that may provide even better results.

Lastly, this form of surface convolution clearly has applications outside the field of prosthetics, as it can be applied to any textured 3D model. It could for instance be used to enhance certain details in landscape photogrammetry, or any number of other applications, even machine learning, though this would require 3D training data, so other methods that work with 2D training data are likely better suited in this case.

5.4 Further Work

Though the image processing in this project did not prove effective enough to be directly applied to creating digital sockets without extensive manual use of CAD software, this is an avenue that should be explored further. Abdelaziz[19] demonstrated with their master's project that partially adjustable prosthetic sockets can be mass produced with 3D printing, and Gorski et al.[20] have successfully tested automatically designed upper limb sockets. We are as yet unaware of a clinical study that has successfully produced a self suspending transradial prosthetic socket using fully digital methods. However, we are also unaware of a clinical trial that has utilised fully-textured 3D scanning and anatomical feature marking in its process. It is our belief that further clinical trials should be carried out with 3D sculpters/CAD engineers working closely with trained prosthetists using fully textured 3D models, attempting to create traditional sockets by replacing the traditional casting process with digital alternatives.

Sockets created with novel methods should ideally be compared to traditional sockets in clinical trials, but the metrics used for evaluating sockets should also be standardised. The socket comfort score should be used in all relevant studies, and a more systematic method for testing and evaluating suspension security should be implemented. More clinical trials involving novel socket production methods should include the use of the socket over a longer period of time, outside of a clinical setting. Adjustable 3D printed sockets should be further investigated and combined with shape capture and automated design. Likewise specialised digital tools should be developed that allow for the rapid design of fully customised adjustable sockets, utilising pre-designed adjustable components.

Adjustable transradial sockets should be further researched in general, as being able to adjust the volume and/or amount of supracondylar compression is likely to increase the performance of the socket in a number of different scenarios. The strength of 3D printed prosthetic sockets has been studied quite extensively already. However, their resistance to long term wear and tear should be studied further, as well as the environmental impact of 3D printed plastic sockets.

Methods of digitally interpreting the surface of textured 3D scans, including the novel convolution method presented in this project should be further explored, with the aim of creating digital design tools with automated features for rapid custom prosthetic design. As the number of clinics using digital shape capture increases, this also increases the viability of machine learning algorithms being used in automated design, though it is unlikely that any form of automation could ever fully replace a human prosthetist, given the intangible factors in prosthetic design discussed earlier. Automation in this context should instead help the prosthetist achieve their vision as easily and quickly as possible, allowing for a faster turnaround and greater level of customisation in each socket.

There is also the possibility to study subsurface scanning, such as MRI in the pursuit of designing more optimal sockets. Though the technology is extremely expensive compared to surface shape capture techniques, it might prove viable for some large clinics with state coverage in countries like Norway, with Ultrasound as a more economical though less accurate option. As real-time scanning on smartphones becomes more widely available, this technology should be evaluated for prosthetics, and the best practices for obtaining high-quality limb captures should be identified.

6 Conclusion

Looking at the existing literature and our own experiments, one thing becomes abundantly clear: no aspect of the socket manufacturing process is trivial, regardless of the tools used. Limb capture, design and modification, and the manufacturing itself, all require some level of expertise, as well as specialised tools. Whilst we were unable to successfully produce viable self suspending transradial sockets, we were lacking expertise in several areas. Still, through improving our skills and knowledge we came reasonably close. Having the fully textured 3D representation of the limb with anatomical markings was incredibly helpful, and contributed greatly to what success we had. The novel method of applying convolution to a textured 3D object which we developed proved effective on a fundamental level, but requires further development if it is to become reliable and applicable to more complex image processing operations.

Emerging digital technologies are already being successfully used by some prosthetists, but the rate of adoption, especially for self suspending upper limb prosthetic sockets, is still slow. A full digitization of the prosthetics field would require enormously extensive re-training, and there is little evidence that such a transition would even be desirable. Optimal patient outcomes should be the primary focus of any medical profession, and the human element is a vital part of the socket manufacturing process. Therefore, digital technologies and automation should be seen as an enhancement to the traditional socket manufacturing process, not a replacement. Nevertheless, there are clear benefits to the integration of digital methods, not only potentially in cost, but also with regards to record keeping, working with patients remotely, knowledge sharing, more rapid innovation, and potentially giving more patients access to self-adjustable

prosthetics that can be better adapted to the wearers daily needs.

References

[1] Li W, Shi P, Yu H. Gesture Recognition Using Surface Electromyography and Deep Learning for Prostheses Hand: State-of-the-Art, Challenges, and Future. Front Neurosci. 2021 Apr 26;15:621885. doi: 10.3389/fnins.2021.621885. PMID: 33981195; PMCID: PMC8107289.

[2] Gallagher, W. (March 05, 2021). How to use the LiDAR scanner in iPhone 12 Pro. Retrieved from https://appleinsider.com/articles/21/03/02/how-to-use-the-lidar-scanner-in-iphone-12-pro

[3] Sha, A. (March 8, 2021). Top 10 Free CAD Software You Should Use. Retrieved from https://beebom.com/free-cad-software/

[4] About Blender. Retrived from https://www.blender.org/about/

[5] Boffey, D. (April 30, 2021). Dutch couple become Europe's first inhabitants of a 3D-printed house. Retrieved from https://www.theguardian.com/technology/2021/apr/30/dutch-couple-move-into-europe-first-fully-3d-printed-house-eindhoven

[6] Forrest, B. (March 28, 2020). Gun-Rights Activist Releases Blueprints for Digital Guns. Retrieved from https://www.wsj.com/articles/gun-rights-activist-releases-blueprints-for-digital-guns-11585414671

[7] Kim, J.H., Seol, YJ., Ko, I.K. et al. 3D Bioprinted Human Skeletal Muscle Constructs for Muscle Function Restoration. Sci Rep 8, 12307 (2018). https://doi-org.ezproxy.uio.no/10.1038/s41598-018-29968-5

[8] J. Olsen, S. Day, S. Dupan, K. Nazarpour and M. Dyson, "3D-Printing and Upper-Limb Prosthetic Sockets: Promises and Pitfalls," in IEEE Transactions on Neural Systems and Rehabilitation Engineering, vol. 29, pp. 527-535, 2021, doi: 10.1109/TNSRE.2021.3057984.

[9] Ten Kate J, Smit G, Breedveld P. 3D-printed upper limb prostheses: a review. Disabil Rehabil Assist Technol. 2017 Apr;12(3):300-314. doi: 10.1080/17483107.2016.1253117. Epub 2017 Feb 2. PMID: 28152642.

[10] Roland K. Chen, Yu-an Jin, Jeffrey Wensman, Albert Shih, Additive manufacturing of custom orthoses and prostheses—A review, Additive Manufacturing, Volume 12, Part A, 2016, Pages 77-89, ISSN 2214-8604, https://doi.org/10.1016/j.addma.2016.04.002. (https://www.sciencedirect.com/science/article/pii/S2214860416300550)

[11] Figliolia, A., Medola, F.O., Sandnes, F.E., Rodrigues, A.C.T. and Paschoarelli, L.C. (2020). Avoiding product abandonment through user centered design: A case study involving the development of a 3D printed customized upper limb prosthesis. In: M. Di Nicolantonio, E. Rossi and T. Alexander (Eds.), Advances in additive manufacturing, modeling systems

and 3D prototyping: Proceedings of the AHFE 2019 International Conference on Additive Manufacturing, Modeling Systems and 3D Prototyping, Cham: Springer (pp. 289-297). doi:10.1007/978-3-030-20216-3-27

[12] Binedell T, Subburaj K, Wong Y, Blessing LTM. Leveraging Digital Technology to Overcome Barriers in the Prosthetic and Orthotic Industry: Evaluation of its Applicability and Use During the COVID-19 Pandemic. JMIR Rehabil Assist Technol. 2020 Nov 5;7(2):e23827. doi: 10.2196/23827. PMID: 33006946; PMCID: PMC7677018.

[13] Sang Y, Li X, Luo Y. Biomechanical design considerations for transradial prosthetic interface: A review. Proc Inst Mech Eng H. 2016 Mar;230(3):239-50. doi: 10.1177/0954411915624452. Epub 2016 Jan 12. PMID: 26759485.

[14] Paternò, Linda and Ibrahimi, Michele and Gruppioni, Emanuele and Menciassi, Arianna and Ricotti, Leonardo. (2018). Sockets for Limb Prostheses: A Review of Existing Technologies and Open Challenges. IEEE Transactions on Biomedical Engineering. PP. 1-1. 10.1109/TBME.2017.2775100.

[15] https://prostheticdesign.com/

[16] Ismail, R.; Taqriban, R.B.; Ariyanto, M.; Atmaja, A.T.; Sugiyanto; Caesarendra, W.; Glowacz, A.; Irfan, M.; Glowacz, W. Affordable and Faster Transradial Prosthetic Socket Production Using Photogrammetry and 3D Printing. Electronics 2020, 9, 1456. https://doi.org/10.3390/electronics9091456

[17] van der Stelt M, Verhulst AC, Vas Nunes JH, Koroma TAR, Nolet WWE, Slump CH, Grobusch MP, Maal TJJ, Brouwers L. Improving Lives in Three Dimensions: The Feasibility of 3D Printing for Creating Personalized Medical Aids in a Rural Area of Sierra Leone. Am J Trop Med Hyg. 2020 Apr;102(4):905-909. doi: 10.4269/ajtmh.19-0359. PMID: 32100676; PMCID: PMC7124926.

[18] Sterkenburg AJ, Van der Stelt M, Koroma AR, Van Gaalen MD, Van der Pols MJ, Grobusch MP, Slump CH, Maal TJJ, Brouwers L. Quality of life of patients with 3D-printed arm prostheses in a rural area of Sierra Leone. Heliyon. 2021 Jul 2;7(7):e07447. doi: 10.1016/j.heliyon.2021.e07447. PMID: 34286125; PMCID: PMC8273216.

[19] Abdelaziz,IMK. The V3D socket: Designing a Volume-adjustable, Affordable, 3D printable, Transradial, Prosthetic socket. http://resolver.tudelft.nl/uuid:68bdb7ad-35a9-48bb-814e-3cc619040698

[20] Górski F, Wichniarek R, Kuczko W, Żukowska M. Study on Properties of Automatically Designed 3D-Printed Customized Prosthetic Sockets. Materials (Basel). 2021;14(18):5240. Published 2021 Sep 12. doi:10.3390/ma14185240

[21] Champagne, N., Eadie, L., Regan, L. et al. The effectiveness of ultrasound in the detection of fractures in adults with suspected upper or lower limb injury: a systematic review and subgroup meta-analysis. BMC Emerg Med 19, 17 (2019). https://doi-org.ezproxy.uio.no/10.1186/s12873-019-0226-5

[22] Kim DI. PREFACE: How Dangerous Are X-ray Studies That We Undertake Every Day?. J Korean Med Sci. 2016;31 Suppl 1(Suppl 1):S2-S3. doi:10.3346/jkms.2016.31.S1.S2

[23] Thompson A., McNally D., Maskery I., Leach R.K. X-ray computed tomography and additive manufacturing in medicine: A review. Int. J. Metrol. Qual. Eng. 2017;8:17. doi: 10.1051/ijmqe/2017015.

[24] Cullen, S., Mackay, R., Mohagheghi, A., and Du, X. (2021). The Use of Smartphone Photogrammetry to Digitise Transtibial Sockets: Optimisation of Method and Quantitative Evaluation of Suitability. Sensors (Basel, Switzerland), 21(24), 8405. https://doi.org/10.3390/s21248405

[25] Kundu,A. et al. Virtual Multi-view Fusion for 3D Semantic Segmentation. arXiv:2007.13138

[26] Graham,B.,Engelcke,M.,van der Maaten,L. 3D Semantic Segmentation with Submanifold Sparse Convolutional Networks. https://arxiv.org/pdf/1711.10275.pdf

[27] Zhou, P., Ye, W., and Wang, Q. (2011). An Improved Canny Algorithm for Edge Detection. Journal of Computational Information Systems, 7(5), 1516-1523.

[28] https://www.einscan.com/handheld-3d-scanner/2x-plus/

[29] Sakshi Indolia, Anil Kumar Goswami, S.P. Mishra, Pooja Asopa, Conceptual Understanding of Convolutional Neural Network- A Deep Learning Approach, Procedia Computer Science, Volume 132, 2018, Pages 679-688, ISSN 1877-0509, https://doi.org/10.1016/j.procs.2018.05.069.

[30] Singh SP, Wang L, Gupta S, Goli H, Padmanabhan P, Gulyás B. 3D Deep Learning on Medical Images: A Review. Sensors (Basel). 2020;20(18):5097. Published 2020 Sep 7. doi:10.3390/s20185097

[31] Suyi Yang E, Aslani N, McGarry A. Influences and trends of various shape-capture methods on outcomes in trans-tibial prosthetics: A systematic review. Prosthet Orthot Int. 2019 Oct;43(5):540-555. doi: 10.1177/0309364619865424. Epub 2019 Jul 31. PMID: 31364475.

Appendix

The code and 3D models from this project will be made available through the following github repository: https://github.com/lothorse/MastersCodeAndExamples.git